ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

RAPPORT DE PROJET DE 9 CRÉDITS
PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE EN GÉNIE LOGICIEL
M.Ing.

PAR
GUY BERTRAND

# RÉINGÉNIERIE D'UN LOGICIEL DE GESTION DE PROCESSUS D'AFFAIRES - MIGRATION DE POSTGRESQL VERS HBASE

MONTRÉAL, LE 10 DÉCEMBRE 2015

**PRÉSENTATION DU JURY**

CE RAPPORT DE PROJET A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

Professeur Alain April, directeur de projet
Département de génie logiciel à l'École de technologie supérieure

Professeure Ghizlane El Boussaidi, membre du jury
Département de génie logiciel et des TI à l'École de technologie supérieure

# RÉINGÉNIERIE D'UN LOGICIEL DE GESTION DE PROCESSUS D'AFFAIRES POUR FAIRE UNE MIGRATION DE POSTGRESQL VERS NOSQL

GUY BERTRAND

## RÉSUMÉ

L'objectif primaire de ce projet est d'accomplir une maintenance évolutive pour modifier la base de données utilisée par un logiciel d'une base de données relationnelle vers une base de données de la technologie BigData. Il y a aussi trois objectifs secondaires associés à ce projet de recherche appliquée. La première est d'utiliser la méthodologie prescrite par les auteurs de «Object-Oriented Reengineering Patterns OORP» (Demeyer, Ducasse, & Nierstrasz, 2002) pour accomplir cette rétro-ingénierie. La deuxième est d'accomplir cette migration sur un logiciel dans le domaine de la gestion des processus d'affaires, connue en anglais sous la rubrique «Business Process Management BPM». Le troisième objectif est d'apprendre une nouvelle technologie de bases de données, le NoSQL du domaine du BigData.

La méthode de planification du cadre de Basili pour les projets de génie logiciel (présentée en annexe) a été utilisée pour ce projet. Ce projet contient les sections suivantes: la première section contient une revue de la littérature portant sur plusieurs sujets : 1) la rétro-ingénierie du logiciel; 2) la gestion des processus d'affaires; et 3) la technologie émergente du NoSQL. La section 2 décrit l'apprentissage de la technologie NoSQL et plus spécifiquement de la technologie HBase. La section 3 décrit les activités de suivi et de journalisation à l'aide de concepts du processus OORP sur un logiciel expérimental : Oryx. Ce logiciel expérimental de BPM est un éditeur graphique qui permet de modéliser des processus d'affaires. Ce logiciel était de source libre et a été originalement créé dans le milieu éducationnel. Le logiciel Oryx utilise actuellement le système de gestion de base de données PostgreSQL. L'objectif de ce projet de recherche est de le remplacer par la technologie NoSQL HBase.

# MIGRATING THE DATABASE SUB-SYSTEM USED BY A BUSINESS PROCESS MANAGEMENT APPLICATION FROM POSTGRESQL TO NOSQL

GUY BERTRAND

## ABSTRACT

The primary objective of this 9 credits graduate applied research project is to perform an evolutionary maintenance activity to replace the relational database sub-system used by an open-source software by a BigData database technologie coined as NoSQL databases.

There are three secondary objectives related to this project. The first is to apply the software reengineering methodology proposed by the authors of "Object-Oriented Reengineering Patterns OORP" (Demeyer, Ducasse, & Nierstrasz, 2002). The second is to perform the reengineering activity on "Business Process Management BPM" open source software: Orynx. The third objective is to learn and apply an emerging technology, specifically a NoSQL technology issue from the emerging BigData domain: HBase.

The software engineering research-planning framework of Victor Basili planning was used (see in attachment) to plan this applied research project. This report, in its first section, presents a literature review of a number of related topics, including: 1) software reverse engineering and maintenance; 2) business process management; and 3) the emerging NoSQL technology. Section two presents the learning process of an emerging NoSQL technology, specifically HBase. The third section describes the steps followed while using the OORP reengineering process. The software used for this case study is Oryx, a graphical editor of business processes. This software was open source at the time and was initially developed by a consortium of universities. Oryx implemented a PostgreSQL database system. The objective of this applied research project is to migrate it to HBase to ensure a better scalability of the software.

## TABLE DES MATIÈRES

Page


INTRODUCTION ..... 1

CHAPITRE 1 Section 1 - REVUE DE LA LITTÉRATURE ..... 3
1.1 Introduction ..... 3

CHAPITRE 2 Revue de la littérature - rétro-ingénierie du logiciel ..... 5
2.1 Rétro-ingénierie ..... 5
2.2 Étendue des critères de recherche ..... 5
2.3 Sommaire/Synthèse des thèmes ..... 5
2.4 Méthodologie ..... 5
2.5 Discussions concernant la rétro-ingénierie ..... 6
2.5.1 La rétro-ingénierie d'un logiciel ..... 6
2.5.2 La réingénierie – l'étape qui suit la rétro-ingénierie ..... 8
2.5.3 Les problèmes de la rétro-ingénierie ..... 9
2.5.3.1 Besoins d'entreprise/affaires ..... 9
2.5.3.2 Connaissances du domaine ..... 9
2.5.3.3 Développent agile ou rapide, sans documentation ..... 10
2.5.3.4 Adaptation d'un système existant pour utiliser de nouvelles architectures ..... 10
2.5.3.5 Perception et attitude envers ce genre de projet ..... 11
2.5.3.6 Problème d'ordre technique dans le développement de logiciels, l'ingénierie et la gestion de ce genre de projet ..... 12
2.5.3.7 Les lois de l'évolution des logiciels ..... 13
2.5.3.8 Le plus gros défi : la compréhension d'un logiciel ..... 13
2.5.3.9 Les stratégies cognitives utilisées par le programmeur pour comprendre un logiciel ..... 15
2.5.3.10 L'utilisation d'outil pour automatiser la compréhension ..... 16
2.5.3.11 Analyses statiques ..... 17
2.5.3.12 Analyses dynamiques ..... 18
2.5.3.13 Visualisation de code ..... 18
2.6 Sommaire des outils ..... 22
2.7 L'éducation de la maintenance logicielle à l'université ..... 23
2.8 Solutions proposées ..... 23
2.9 Rhétorique – les obstacles de la rétrotraçabilité ..... 24

CHAPITRE 3 Revue de la littérature Gestion des processus d'affaires ..... 25
3.1 Introduction ..... 25
3.2 Étendue des critères de recherche ..... 25
3.3 Sommaire / Synthèse des thèmes ..... 25
3.4 Méthodologie ..... 26

3.5 Discussions sur GPA (BPM) ....26
3.6 Définitions....29
3.7 Prochaine évolution : le GPA intelligent (iBPM) ....33

CHAPITRE 4 Revue de la littérature BigData ....37
4.1 Définition et description du problème ....37
4.2 Les outils BigData pour ce projet ....38
4.3 Pourquoi utiliser les « mégadonnées »....39
4.4 Pourquoi ne pas utiliser BigData ....40
4.5 Le succès avec les écosystèmes BigData....41
4.6 La quantité et la qualité des données. Sont-elles compatibles? ....41
4.6.1 La qualité des données .... 41
4.6.2 Agir correctement sur cette quantité de données .... 42
4.7 Les composantes requises pour traiter les mégadonnées....43
4.8 Et évidemment, la quantité ou le volume de données....44
4.9 L'architecture Big Data ....44

CHAPITRE 5 Revue de la littérature – Il faut mentionner le « cloud »....45
5.1 Infonuagique, le nuage, le « cloud »....45
5.2 Infonuage : menace au directeur de système d'information? ....46
5.3 L'infonuage…ce n'est qu'un serveur de quelqu'un d'autre… ....46
5.4 Exemple vécu....47
5.5 Leçon apprise ....49

CHAPITRE 6 Revue de la littérature – Utilisation de BPM et BigData ensemble ....51
6.1 Interaction entre BPM et Big Data....51

CHAPITRE 7 Section 2 - Apprentissage de la technologie BigData ....53
7.1 Introduction – cadre de Basili ....53
7.2 Vidéos éducationnels sur Youtube.com....53
7.3 Tutoriel #1 - Première installation d'un environnement complet - Cloudera ....55
7.3.1 Cloudera Express .... 55
7.3.2 Un petit sommaire de l'entreprise Cloudera et ses produits .... 56
7.3.3 Formation offerte .... 57
7.3.4 Survol de la machine virtuelle « Express »....59
7.3.5 Logiciels BigData installés sur la machine virtuelle....61
7.3.6 Tutoriel de Cloudera .... 63
7.3.6.1 Tutoriel section #1 - Intégration avec des systèmes existants de base de données et l'ingestion des données existante.... 63
7.3.6.2 Tutoriel section #2 - Interrogation des données structurées .... 65
7.3.6.3 Tutoriel section #3 - Relié les données structures avec les données non structurées .... 66
7.3.6.4 Tutoriel section #4 - Analyses avancées avec l'utilisation de « Spark » .... 67

7.3.6.5 Appréciation du tutoriel de Cloudera ..... 68
7.4 Tutoriel #2 – Installation de HBase ..... 68
7.4.1 Apache HBase ..... 69
7.4.2 Attention aux néophytes! ..... 69
7.4.3 Tutoriel choisi – HBase, installation locale et unique sur un ordinateur portatif avec Windows 7 ..... 70
7.4.4 Les prérequis ..... 70
7.4.5 Installation de HBase ..... 72
7.4.6 Installation et configuration de SSH ..... 73
7.4.7 Configuration de HBase ..... 74
7.4.8 Winutils ..... 75
7.4.9 Échec… ..... 75
7.4.10 Utilisation des fichiers CMD au lieu de SH ..... 76
7.4.11 Première aperçue des fichiers CMD ..... 76
7.4.12 Une dernière modification ..... 77
7.4.13 HBase en action! ..... 78
7.4.14 Vérification de l'installation de HBase ..... 78
7.4.15 Reculer d'un pas ..... 80
7.4.16 Une autre approche plus facile ..... 80
7.5 Tutoriel #3 – un programme Java avec HBase ..... 80
7.5.1 Exemple #1 dans le blogue de Xu Fei ..... 81
7.5.2 Les librairies (*.jar) ..... 81
7.5.3 Première exécution de l'exemple #1 ..... 83
7.5.4 Exemple #2 dans le blogue de Xu Fei ..... 86
7.5.5 MapReduce Tutorial – Hadoop.apache.org ..... 87

CHAPITRE 8 Section 3 – Application de technique de rétro-ingénierie dans le contexte de technologies BigData ..... 91
8.1 Introduction ..... 91
8.2 « Object-Oriented Reengineering Patterns – OORP » ..... 91
8.3 Le cycle de vie de la rétro-ingénierie ..... 92
8.4 Le projet Oryx ..... 95
8.5 Le problème avec Oryx ..... 96
8.6 Objectif de modification pour Oryx ..... 96
8.7 OORP avec Oryx ..... 96
8.7.1 Groupement patrons : Établir la direction « setting direction » ..... 96
8.7.1.1 Établir un accord de principe et règle de conduite (Agree on Maxims) ..... 97
8.7.1.2 Nommé un navigateur (Appoint a Navigator) ..... 98
8.7.1.3 La communication dans l'équipe (Speak to the round table) ..... 98
8.7.1.4 Établir la priorité des problèmes (Most Valuable First) ..... 98
8.7.1.5 Correction des problèmes et non des symptômes (Fix Problems, Not Symptoms) ..... 99
8.7.1.6 Corriger que les vrais problèmes (If it ain't broke, don't fix it) 99

8.7.1.7 La simplicité (Keep it simple)..... 99
8.7.2 Groupement de patrons : Première aperçue (First Contact)..... 100
8.7.2.1 Discussion avec l'équipe de maintenance (Chat with the mainteners)..... 101
8.7.2.2 Faire une révision du code en une heure (Read all the code in one Hour)..... 101
8.7.2.3 Faire un survol rapide de la documentation « Skim the Documentation »..... 104
8.7.2.4 Visionnez une démonstration et posez des questions ( Interview during Demo)..... 105
8.7.2.5 Faire une installation test (Do a Mock Installation)..... 105
8.7.3 Groupement patrons : Premier survol de l'application (Initial Understanding)..... 106
8.7.3.1 Faire une analyse du niveau de la persistance (Analyse the Persistent Data)..... 107
8.7.3.2 Créer des hypothèses sur le design de l'application (Speculate about Design)..... 111
8.7.3.3 Découvrir les problèmes potentiels de design (Study the Exception Entities)..... 112
8.7.4 Groupement de patrons : Acquisition des détails du modèle (Detailed Model Capture)..... 113
8.7.4.1 Faire le lien entre vos questions et le code (Tie Code and Questions)..... 114
8.7.4.2 Réingénierie logicielle pour comprendre (Refactor to Understand)..... 115
8.7.4.3 Naviguer à travers l'exécution (Step throught the Execution). 115
8.7.4.4 Trouver les contrats (Look for the Contracts)..... 116
8.7.4.5 Apprendre de l'historique (Learn from the past)..... 116
8.7.5 Exécution de la rétro-ingénierie..... 117
8.7.5.1 Les tests : absolument nécessaire (tests: your life insurance).. 117
8.7.5.2 Créer des tests pour permettre l'évolution (Write Tests to Enable Evolution)..... 118
8.7.5.3 Ajouter à vos tests par étape (Grow your test base incrementally)..... 119
8.7.5.4 Utiliser un jeu de tests (Use a testing framework)..... 119
8.7.5.5 Vérifier le comportement et non son implémentation (Test the Interface, Not the Implementation)..... 119
8.7.5.6 Écrire les tests selon les règles d'affaires (Record Business Rules as Tests)..... 120
8.7.5.7 Écrire des tests pour comprendre le comportement de l'application (Write Tests to Understand)..... 120
8.7.5.8 Stratégie de migration (Migration Strategies)..... 120
8.7.5.9 Impliquer les usagers (involve the users)..... 121

8.7.5.10 Bâtir la confiance dans le processus (Build Confidence) ..... 121
8.7.5.11 Migration par étape (Migrate Systems Incrementally) ..... 122
8.7.5.12 Créez un prototype (Prototype the Target Solution)..... 122
8.7.5.13 Faire la construction du logiciel de façon régulière (Always have a running version) ..... 122
8.7.5.14 Les tests de régression (Regression test after every change)... 122
8.7.5.15 Utilisation des 2 logiciels en parallèle avec un pont (Make a bridge to the new town) ..... 123
8.7.5.16 Écrire une couche isolante autour de l'ancienne application (Present the right interface)..... 123
8.7.5.17 Établir deux niveaux d'interface (Distinguish public from published interface)..... 123
8.7.5.18 Annoncez les interfaces désuètes (Deprecate obsolete interfaces) ..... 123
8.7.5.19 Évitez les changements trop brusques (Conserve familiarity). 123
8.7.5.20 Identifiez les goulots d'étranglement (Use profiler before optimizing)..... 123
8.7.5.21 La duplication de code (Detecting duplicated code)..... 124
8.7.5.22 Comparez le code source (Compare code mechanically) ..... 124
8.7.5.23 Visionner le code avec un graphique (Visualize code as dotplots) ..... 124
8.7.5.24 La redistribution des responsabilités (redistribute responsibilities) ..... 125
8.7.5.25 Rapatriez la logique plus proche des données (Move behavior close to data) ..... 125
8.7.5.26 Élimination du code qui ne fait que de la direction (Eliminate Navigation code)..... 126
8.7.5.27 La classe à tout faire (Split up God class)..... 126
8.7.5.28 Réduisez l'utilisation des classes avec de grosses séquences de décision (transform conditionals to polymorphism)..... 126
8.7.5.29 Remplacement du code conditionnel par un système de "hooks" (Transform self type checks) ..... 127
8.7.5.30 Réduisez la vérification du genre de classes (Transform client type checks) ..... 127
8.7.5.31 Élimination de la vérification de l'état (Factor out state) ..... 128
8.7.5.32 Élimination du choix d'algorithme (Factor out strategy) ..... 128
8.7.5.33 Élimination du code qui vérifie pour la valeur "null" (Introduce null object) ..... 128
8.7.5.34 Utilisez un système d'enregistrement (Transform conditionals into registration)..... 128

CHAPITRE 9 Hibernate – et l'impact sur mon projet.....129
9.1 Introduction.....129
9.2 JDBC - définition.....129

9.3 Hibernate - définition ..... 130
9.4 Hibernate – avantages et défis ..... 131
9.5 En prévision de la conversion JDBC avec Oryx ..... 132

CONCLUSION ..... 135

ANNEXE I Tutoriel #3 – example #1 de Xu Fei ..... 139

ANNEXE II Tutoriel #3 – exemple #2 de Xu Fei ..... 141

ANNEXE III Cadre de Basili ..... 145
9.6 Définition ..... 145
9.7 Planification ..... 146
9.8 Exécution/Développement ..... 147
9.9 Interprétation ..... 147

ANNEXE IV Oryx – schéma de base de données ..... 149

ANNEXE V Oryx – résultat de l'exécution du script db_schema.sql ..... 173

LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES ..... 179

Travaux cités ..... 179

## LISTE DES FIGURES

Page

Figure 1 - le modèle en fer à cheval (Yu, et al., 2005) ....8

Figure 2 - Outil Rigi....19

Figure 3 - Outil Rigi....19

Figure 4 - outil SHRIMP....20

Figure 5 - outil SHRIMP....20

Figure 6 - outil Understand....21

Figure 7 - outil IBM Rational ....21

Figure 8- L'Infonuagique n'existe pas...vous utilisez l'ordinateur d'autrui (twitter.com, 2015) ....47

Figure 9 - capture-écran Hyper-V....60

Figure 10 - capture d'écran - VMWare ....60

Figure 11 - capture d'écran Cloudera....61

Figure 12 - schéma base de données tutoriel Cloudera (Cloudera) ....64

Figure 13 - capture d'écran - Cloudera query editors....65

Figure 14 - capture d'écran - Cloudera data browsers....65

Figure 15 - capture d'écran - Cloudera Impala ....66

Figure 16 - capture d'écran - Cloudera Flume ....67

Figure 17 - capture d'écran - Cloudera Spark ....68

Figure 18 - capture d'écran - installation Java ....71

Figure 19 - capture d'écran - Cygwin....71

Figure 20 - capture d'écran - SSH....72

Figure 21 - capture d'écran – diff....72

Figure 22 - capture d'écran - HBase....73

Figure 23 - capture d'écran - SSH configuration ....73

Figure 24 - capture d'écran - config HBase 1 ....74

Figure 25 - capture d'écran - config HBase 2 ....74

Figure 26 - capture d'écran - HBase configuration 3 ....75

Figure 27 - capture d'écran - liste partielle librairies HBase....82

Figure 28 - capture d'écran - projet Eclipse librairies....83

Figure 29 - capture d'écran - premier ajout de librairie ....83

Figure 30 - capture d'écran - première série d'erreurs....84

Figure 31 - capture d'écran - la liste des librairies ....84

Figure 32 - capture d'écran - création d'une BD dans HBase ....85

Figure 33 - capture d'écran - hbase-site.xml ....85

Figure 34 - capture d'écran - programme avec succès ....86

Figure 35 - capture d'écran - classes dans exemple #2 ....86

Figure 36 - capture d'écran - exemple #2 succès ....87

Figure 37 - capture d'écran - map reduce succès ....88

Figure 38 - capture d'écran - MapReduce donne fichier SUCCESS ....89

Figure 39 - capture d'écran - fichiers pour MapReduce....89

Figure 40 - Exemple d'un patron de rétro-ingénierie (Demeyer, Ducasse, & Nierstrasz, 2002) ....94

Figure 41 - patron organisé en groupe dans le cycle de vie (Demeyer, Ducasse, & Nierstrasz, 2002) ....95

Figure 42 - Oryx définition (Oryx-Editor, s.d.) ....95

Figure 43 - OORP Groupement patrons établir direction (Demeyer, Ducasse, & Nierstrasz, 2002) ....97

Figure 44 - Groupement patrons "Première aperçue" (Demeyer, Ducasse, & Nierstrasz, 2002) ..... 101

Figure 45 - capture d'écran - identification des classes Java qui utilisent Hibernate ..... 103

Figure 46 - capture d'écran - le client autonome "thick client" ..... 103

Figure 47 - capture d'écran - structure typique d'une installation Tomcat ..... 104

Figure 48 - Question suspecte (Oryx-Editor, s.d.) ..... 105

Figure 49 – capture d'écran - compilation avec succès ..... 105

Figure 50 - Groupement patrons - premier survol (Demeyer, Ducasse, & Nierstrasz, 2002) ..... 107

Figure 51 - schéma de la BD selon le site web (Oryx-Editor, s.d.) ..... 108

Figure 52 – capture d'écran - création de l'usager POEM et la BD=Poem ..... 109

Figure 53 - capture d'écran - les fonctions or procédure stockée ..... 109

Figure 54 - capture d'écran - les séquences ..... 110

Figure 55 - capture d'écran - les tables ..... 110

Figure 56 - capture d'écran - et un déclencheur automatique (trigger) ..... 111

Figure 57 - capture d'écran - des commentaires dans le code source ..... 113

Figure 58 - groupement de patrons - Acquisition des détails du modèle (Demeyer, Ducasse, & Nierstrasz, 2002) ..... 114

Figure 59 - Regroupement patrons - les tests ..... 118

Figure 60 - regroupement de patrons - la migration (Demeyer, Ducasse, & Nierstrasz, 2002) ..... 121

Figure 61 - regroupement de patrons - détections de duplicata de code ..... 124

Figure 62 - regroupement de patrons - redistribution des responsabilités (Demeyer, Ducasse, & Nierstrasz, 2002) ..... 125

Figure 63- regroupement de patrons - Conversion vers le polymorphisme (Demeyer, Ducasse, & Nierstrasz, 2002) ..... 127

## LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

| | |
|---|---|
| *nix | - dénotation pour indiquer les systèmes d'exploitation Unix ou Linux |
| API | - Application Program Interface |
| Active Directory | - service d'annuaire de Microsoft |
| BD | – base de données |
| BPM | – Business Process Management |
| CD | – anglais: compact disque, français : disque compact |
| ERP / PGI | - Progiciel de gestion intégré – anglais : Enterprise Resource Planning |
| GPA | – Gestion des processus d'affaires |
| HBase | - système de gestion de bases de données non relationnelles distribuées |
| HDFS | - Hadoop Distributed File System |
| Intergiciel | – anglais : middleware |
| JDBC | - « Java Database Connectivity |
| LDAP | – service d'annuaire. Anglais : Light Directory Access Protocol |
| Log4J SFL4j | – librairie utilisé en Java pour la journalisation |
| Mantis | – système de gestion de bogue de source libre |
| OORP | – Object Oriented Reengineering Patterns |
| ORM | – Object Relational Mapping |
| PostgreSQL | - système de gestion de base de données |
| MySQL | - système de gestion de base de données |
| Python | – langage de programmation |
| SAP | – logiciel de PGI (ERP) |
| SGBD | – système de gestion de base de donnée |
| SQL | – Structured Query Language |
| SubVersion | – outils de gestion de version de code source |
| TAR | – outil pour archivage – anglais: Tape ARchiver |
| TI | – technologie de l'information |
| WAR | – fichier d'archive utilisée par le logiciel serveur web Tomcat |
| Workflow | – processus d'affaires |
| XML | – Extensible Markup Language |
| *ZIP | – outils pour compression de fichier (zip, unzip, gzip, gunzip) |
| iBPM | – Intelligent Business Process Management |

# INTRODUCTION

Il y a trois motivations pour ce projet. La première est d'approfondir les connaissances dans la rétro-ingénierie de logiciel. Nous retrouvons souvent la mention que la maintenance de logiciel représente de 60 à 90% des coûts dans le cycle de vie d'un logiciel. Le praticien commençant sa carrière, en génie logiciel, participera rarement dans un nouveau projet de développement logiciel. La majorité des efforts de celui-ci sera concentrée sur la maintenance logicielle, du genre évolutif, c'est-à-dire l'ajout de fonctionnalité à un logiciel existant. La compréhension du logiciel existant devient un enjeu et une étape importante dans ce processus.

La deuxième motivation de ce projet de recherche est d'apprendre une nouvelle technologie moderne NoSQL du domaine du BigData. Cette technologie est décrite comme ayant un grand écosystème (c.-à-d. un nombre élevé de nouvelles technologies) qui permet de traiter une quantité massive de données.

La troisième motivation est d'approfondir les connaissances dans le domaine de la gestion des processus d'affaire, connue en anglais sous le nom «Business Process Management BPM».

Le but de ce projet est d'effectuer la migration du système (c.-à-d. la réingénierie) du module de gestion de la base de données PostgreSQL, qui est utilisé par le logiciel libre Oryx, par une base de données NoSQL, plus spécifiquement la base de données HBase, qui est utilisé dans les nouvelles technologies du domaine émergent du BigData. Pour se faire, une expérimentation d'un processus de rétro-ingénierie sera effectuée.

Ce projet de recherche appliquée, de neuf crédits, est effectué selon la perspective d'un gestionnaire et architecte d'application d'expérience.

Dans la première section, il y aura une revue de la littérature sur plusieurs sujets incluant les systèmes de gestion de processus d'affaires, la rétro-ingénierie et BigData. La deuxième

section contiendra les détails de l'apprentissage de la technologie BigData. Ceci inclura plusieurs étapes, incluant l'installation d'une base de données BigData, pour ensuite suivre plusieurs tutoriels qui sont disponibles en ligne. La dernière étape sera le développement de plusieurs applications pour démontrer l'accès d'une base de données NoSQL. La troisième section sera la rétro-ingénierie du logiciel Oryx pour la faire fonctionner avec une base de données NoSQL en utilisant l'approche « Object-Oriented Reengineering Patterns OORP ».

La dernière section de ce projet présentera les leçons apprises, les résultats de l'effort et une brève description de l'utilisation de ces nouvelles compétences et connaissances dans le futur.

# CHAPITRE 1

## Section 1 - REVUE DE LA LITTÉRATURE

### 1.1 Introduction

Les sujets abordés par cette revue de la littérature sont :

- La rétro-ingénierie et plus précisément la compréhension d'un logiciel existant;
- La gestion des processus d'affaires (GPA), aussi connue sous le nom «Business Process Management (BPM) »; et
- Le BigData et la technologie émergente du NoSQL.

Chaque sujet sera présenté dans son propre chapitre. De plus, il y aura un chapitre additionnel au sujet de l'utilisation et l'interaction entre GPA et Big Data.

# CHAPITRE 2

## Revue de la littérature - rétro-ingénierie du logiciel

### 2.1 Rétro-ingénierie

Cette revue de la littérature se concentre sur l'aspect de la compréhension d'un logiciel, une étape importante et colossale dans le processus de maintenance logicielle et rétro-ingénierie.

### 2.2 Étendue des critères de recherche

Lors du processus de recherche, les termes suivants ont été utilisés, en français et anglais:

- Maintenance logicielle - Software maintenance
- Rétro-ingénierie - Software reverse engineering
- Archéologie logiciel - Software archeology
- Compréhension logiciel - Software comprehension

### 2.3 Sommaire/Synthèse des thèmes

- Le plus gros défi : la compréhension d'un logiciel
- Les stratégies cognitives utilisées par le programmeur pour comprendre un logiciel
- L'utilisation d'outil pour automatiser la compréhension
- L'éducation de la maintenance logicielle à l'université et en entreprise

### 2.4 Méthodologie

Il est important de noter que cette revue est ciblée vers un aspect particulier de la rétro-ingénierie. On discute de la compréhension d'un logiciel.

De plus, les termes suivants seront utilisés de façon interchangeable pour aider à la compréhension et à l'écriture :

- Programmeur, développeur, ingénieur logiciel
- Logiciel, système informatique, programme, code source

## 2.5 Discussions concernant la rétro-ingénierie

Les questions suivantes représentent le défi journalier d'un développeur logiciel :

- Vous avez une journée pour ajouter une nouvelle fonctionnalité dans un logiciel de 34000 lignes. Où commencer?
- Comment comprendre et simplifier une section de code incompréhensible?
- Est-il possible de détisser un processus de compilation compliqué?
- Comment comprendre une section de code qui semble faire 5 choses en parallèle?

(traduction libre (Spinellis, 2003))

### 2.5.1 La rétro-ingénierie d'un logiciel

La rétro-ingénierie est le processus d'analyse d'un système existant pour retrouver de l'information perdue ou générer de l'information manquante. Ce processus identifie les composantes de ce système, les relations entre ces composantes, pour ensuite créer une ou plusieurs représentations de ce système dans un autre format ou dans un modèle avec un niveau d'abstraction plus élevé (Demeyer, Ducasse, & Nierstrasz, 2002) (Riva, Selonen, Systa, & Xu, 2004) (Sadiq & Waheed, 2011). On peut le décrire comme un processus pour recréer de documentation (Sadiq & Waheed, 2011).

La rétro-ingénierie est utilisée pour corriger plusieurs types de problèmes. Ceci inclut la compréhension d'un système existant ainsi que la récupération de l'information qui a été égarée (Kienle & Müller, 2010). Les départs d'employées, un manque de discipline ou de rigueur dans la gestion de la vie d'un logiciel sont tous des facteurs qui aggravent la perte d'information.

Plusieurs techniques peuvent être utilisées pour aider dans le processus de rétro-ingénierie, surtout pour réduire l'impact de la complexité d'un système existant. On peut utiliser plusieurs artéfacts existants, tels que :

- Entrevue et discussion avec des collègues qui ont déjà travaillé sur le système
- Code source et gestionnaire de versions de code source (commentaires, journaux des modifications). Parfois, le code source est la seule source d'information (Canfora, Di Penta, & Cerulo, 2011) (Storey, Fracchia, & Müller, Cognitive design elements to support the construction of a mental model during software exploration, 1999)
- Documents existants
    - D'architecture, exigences des usagers, autres documents de conception
    - manuels d'utilisation et de formation pour les usagers
    - connaissances du domaine, connaissances générales du problème et solutions possibles
    - expérience personnelle des membres de l'équipe

La rétro-ingénierie peut être un objectif en lui-même. L'objectif primaire de ce genre de projet est de recréer un niveau de confort et d'apprentissage pour une équipe de support, ou de maintenance logicielle. Ceci devient nécessaire à la suite de départ de ressources clés pour la retraite ou d'autre emploi.

La rétro-ingénierie peut être appliquée à un système patrimonial, possiblement vieux de plus de 15 années, écrit dans un langage de troisième génération. On pense rapidement à l'utilisation du langage de programmation C ou le Cobol avec un style de programmation procédurale. Cependant, il ne faut pas oublier que la rétro-ingénierie peut aussi s'appliquer à une jeune application écrite avec un langage orienté objet. L'information peut être perdue à n'importe quelle étape du cycle de vie d'un système logiciel.

### 2.5.2 La réingénierie – l'étape qui suit la rétro-ingénierie

Pour la majorité des projets de rétro-ingénierie, la continuation logique est un projet de réingénierie. La réingénierie va plus loin que la rétro-ingénierie, car elle consiste à faire une analyse d'un système existant (rétro-ingénierie) pour le recréer dans un nouveau format (l'ingénierie de l'avant) (Demeyer, Ducasse, & Nierstrasz, 2002).

La réingénierie d'un système logiciel est aussi représentée par le modèle « fer à cheval ». Ce modèle est plus spécifique au domaine de l'ingénierie des spécifications. Le modèle « fer à cheval » décrit les étapes de rétro-ingénierie vers l'ingénierie de l'avant, à travers les différentes couches d'un système informatique (c.-à-d. code source, structure du code, fonctions, architecture). On reverra ce genre d'approche dans la section 3 de ce document.

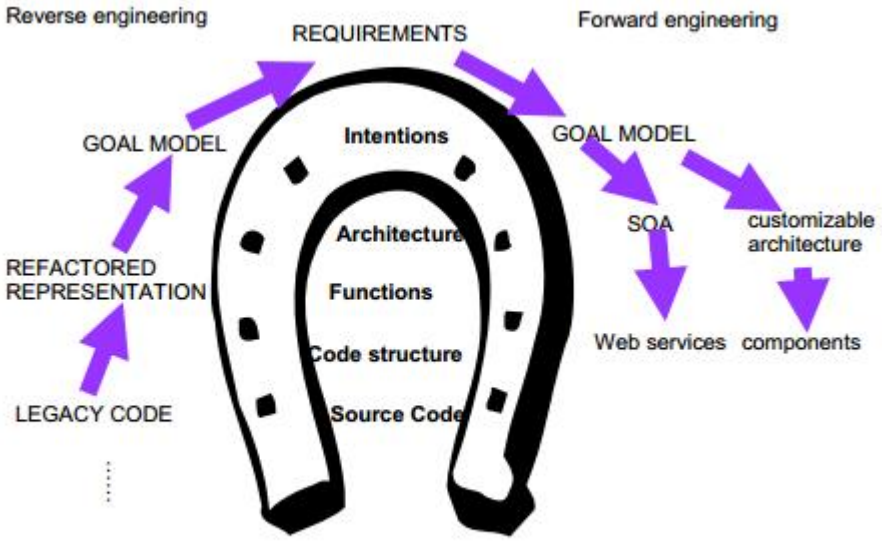


**Figure 1. The horseshoe model**

Figure 1 - le modèle en fer à cheval (Yu, et al., 2005)

### 2.5.3 Les problèmes de la rétro-ingénierie

Il existe plusieurs obstacles pour justifier, démarrer, maintenir ou accomplir un projet de rétro-ingénierie, tel que :

- Besoins d'entreprise/affaires
- Perception et attitude envers ce genre de projet
- Problème d'ordre technique dans le développement de logiciels, l'ingénierie et la gestion de ce genre de projet
- Les lois de l'évolution des logicielles

#### 2.5.3.1 Besoins d'entreprise/affaires

Dans le cycle de vie d'un système logiciel, il est possible que celui-ci occasionne des malaises (financier, gestion, opérationnelle) à une entreprise, car il ne satisfait plus les besoins évolutifs de celle-ci.

L'entreprise doit étudier le choix de complètement remplacer ce système (achat nouveau logiciel ou développement à zéro) ou d'entreprendre un projet de rétro-ingénierie. Les deux choix ont des risques et des coûts qui sont parfois difficiles à justifier, mais parfois douloureusement nécessaires (Booch, 2008).

Défi : il faut vivre avec les contraintes budgétaires de l'entreprise et tenter de découvrir une façon de soulager les douleurs évolutionnaires.

#### 2.5.3.2 Connaissances du domaine

Un logiciel ou un système patrimonial contient une quantité énorme de connaissances du domaine d'affaire et d'historique de l'évolution du système en question.

Il est important de conserver ce genre d'information pour l'utiliser lors du développement du nouveau système (nouveau ou rétro). Cependant, il est souvent très difficile de faire une migration de cette information vers le projet de développement logiciel de remplacement sans perdre de l'information.

Défi : il ne faut pas perdre l'expertise, les connaissances et surtout les exigences usagers qui sont parfois cachées dans un système patrimonial.

#### 2.5.3.3 Développent agile ou rapide, sans documentation

Il existe aussi beaucoup de problèmes de manquement à la documentation de système causé par le manque de rigueur dans l'utilisation d'un processus de développement de genre « agile » (Sadiq & Waheed, 2011). Ce genre de développement logiciel est souvent perçu comme étant un processus qui n'utilise aucune documentation. « la documentation nous ralentit… »

Ceci est faux. Le développement agile utilise la documentation d'une façon différente.

- Le développement en cascade demande que la documentation d'une étape soit complétée avant que la nouvelle étape puisse débuter.
- Le développement agile spécifie que la documentation soit créée pendant ou vers la fin du projet de façon évolutive pour décrire ce qui a été accompli.

Défi : surmonter le manque de documentation utile, causé par un manquement des pratiques de génie logiciel, mauvaise gestion, paresse ou du développement agile mal pratiqué.

#### 2.5.3.4 Adaptation d'un système existant pour utiliser de nouvelles architectures

Depuis quelques années, le domaine de l'architecture applicatif offre de nouveau paradigme et offre maintenant de nouveaux types d'architecture applicatifs, tel que :

- Orienté service (web service)

- Composantes modulaires

Les nouvelles architectures offrent la notion de réutilisation, un aspect qui devient de plus en plus désirable dans les architectures modernes.

Les applications plus âgées ont tendance à offrir un défi de taille pour s'adapter à ce genre d'architecture, car il est plus probable qu'ils utilisent une architecture plus monolithique.

De plus, il est possible qu'un sentiment de frustration soit ressenti par les ingénieurs et développeurs logiciels qui sont plus jeunes ou se recycle régulièrement, car ils ne pourront pas utiliser une architecture plus moderne, ou trouveront que les efforts d'adaptation d'un logiciel âgé vers une nouvelle architecture soient un défi de taille.

Défi : l'adaptation d'un logiciel patrimonial pour utiliser des architectures applicatives plus modernes.

#### 2.5.3.5 Perception et attitude envers ce genre de projet

Il existe une perception négative au sujet des projets de rétro-ingénierie. Ce genre de projet est considéré comme étant ennuyeux, monotone et une activité technique compliquée et de longues haleines. On voit souvent l'expression « si ce n'est pas cassé, ne la répare pas » (de l'anglais : if it ain't broke, don't fix it).

Il n'est pas intéressant de travailler dans le désordre des autres, surtout si :

- Il n'y avait aucune utilisation des pratiques de génie logiciel, tel qu'une documentation qui est à jour,
- manque d'expertise des membres de l'équipe précédente,
- aucune explication des décisions de design ou de programmation
    - pourquoi on il fait ceci de cette façon?
    - Quel est l'objectif de cette section de code?

Défi : changement des pratiques de génie logiciel pour améliorer la qualité du code, avec l'objectif de rendre la maintenance plus facile et moins onéreuse.

#### 2.5.3.6 Problème d'ordre technique dans le développement de logiciels, l'ingénierie et la gestion de ce genre de projet

Il existe plusieurs facteurs qui impactent la capacité de bien faire la rétro-ingénierie d'un logiciel patrimonial, tel que :

- Changement non documenté dans le code qui affecte l'architecture (Riva, Selonen, Systa, & Xu, 2004),
- Changement dans le calibre des développeurs ou utilisation du mauvais genre de développeurs. Possiblement quelqu'un qui n'a aucune expertise de ce domaine ou un « hacker intelligent ». Le terme « hacker intelligent » est utilisé pour décrire un programmeur qui est intelligent, mais qui n'a pas de notions de rétro-ingénierie. Il n'écrit pas son code pour être facilement compris et modifié dans le futur,
- La grosseur de l'application continue d'augmenter, possiblement hors contrôle,
- Les modifications et ajouts peuvent causer une glissade du design original,
- Système de mentorat qui n'est pas en place,
- Roulement des membres de l'équipe, surtout ceux qui sont les seuls à posséder de l'information sur un sujet. Il est possible qu'ils n'aient pas été remplacés (au niveau des connaissances),
- Les technologies utilisées pour le développement du logiciel et l'âge sont des facteurs aggravants,
- Problème de documentation, possiblement causé par des compétences limitées ou des problèmes de manque de temps ou de ressource lors de l'étape de développement (Canfora, Di Penta, & Cerulo, 2011),
- Modification de l'architecture sournoise, non documentée et non intentionnelle, possiblement par les activités de maintenance (Kazman & Carrière, 1999),

- Processus de construction (build) massif, complexe et monolithique au lieu d'être modulaire et simple de lecture (Adams, Tromp, De Schutter, & De Meuter, 2007),
- Même s'il y a une documentation, il est nécessaire de valider qu'elle est encore précise et à jour. Typiquement, il faut utiliser un processus avec plusieurs itérations pour découvrir les correspondances entre le code source et tout genre de documentation. (Riva, Selonen, Systa, & Xu, 2004).

La rétro-ingénierie est un travail monotone et fastidieux et pourtant une activité technique complexe (Akkiraju, Mitra, & Thulasiram).

Défi : Augmenter la gestion du changement et de configuration

#### 2.5.3.7 Les lois de l'évolution des logiciels

Toute discussion de la rétro-ingénierie doit mentionner les lois de Lehman au sujet de l'évolution des logiciels (Demeyer, Ducasse, & Nierstrasz, 2002) (M.M. Lehman, 1997), particulièrement les suivantes :

- La loi sur le changement de logiciel stipule qu'un logiciel doit évoluer sinon il perdra son utilité et sa pertinence,
- La loi sur l'augmentation de la complexité stipule qu'un système devient de plus en plus complexe avec les modifications et ajouts.

Défi : Faire évoluer un logiciel en minimisant les augmentations de la complexité.

#### 2.5.3.8 Le plus gros défi : la compréhension d'un logiciel

Comme mentionné plus tôt, le plus gros défi dans un processus de rétro-ingénierie est la compréhension d'un logiciel existant (Sadiq & Waheed, 2011) (Alnusair & Zhao, 2010)

(O'Brien, 2003). C'est le rôle primaire de la construction et la maintenance d'un logiciel. Il est essentiel de comprendre un logiciel pour bien accomplir ce rôle.

Il existe beaucoup d'obstacles dans le processus de compréhension d'un logiciel, tel que :

- Avoir assez de connaissances du domaine,
- Il faut comprendre la façon dont le logiciel est utilisé incluant ses contraintes, son opération normale et sa liste des problèmes à corriger (s'il y en a) (Demeyer, Ducasse, & Nierstrasz, 2002),
- Il faut comprendre les raisons des modifications. Il est possible de découvrir les intentions du développeur précédent lors de la lecture du code. Il faut découvrir les surprises ou les sections de code douteuses (code smells), ou les anti-patrons de conception,
- Il est à noter que des études ont découvert que les « code smells » ont beaucoup moins d'impact sur la compréhension d'un logiciel que prévu. Il est même possible qu'un « code smell » puisse être utilisé comme point de repère lors de l'étape de compréhension (Abbes, Khomh, Guéhéneuc, & Antoniol, 2011) (Sjoberg, Yamashita, Anda, Mockus, & Dyba, 2013),
- Quand un nouveau programmeur se joint à une équipe (Cubranic & Murphy, 2003) (O'Brien, 2003), il doit :
    - Comprendre la structure organisationnelle,
    - Comprendre le processus de développement logiciel, et les manquements à ce processus,
    - Comprendre la technologie utilisée, et possiblement le cadre de travail utilisé (framework),
    - Comprendre le processus de compilation et de construction du logiciel (modulaire, étapes multiples, etc.).

Comme mentionné, la compréhension d'un logiciel est l'étape la plus importante et la plus longue dans le processus de rétro-ingénierie.

Défi : améliorer les pratiques du génie logiciel, parrainage et possiblement revue de code informel.

#### 2.5.3.9 Les stratégies cognitives utilisées par le programmeur pour comprendre un logiciel

La lecture de code source nécessite un ensemble de compétences particulières. Pour créer un modèle mental d'un logiciel, un développeur utilise plusieurs modèles cognitifs ou stratégies pour l'aider dans le processus de compréhension (Storey, Wong, & Muller, How do program understanding tools affect how programmers understand programs?, 1997) (Storey, Fracchia, & Müller, Cognitive design elements to support the construction of a mental model during software exploration, 1999) (O'Brien, 2003). Certains de ces modèles sont :

- Lecture de code en remontant vers les exigences usagers et spécifications d'architecture (bottom-up) (O'Brien, 2003),
- Lecture des exigences usagers et spécifications d'architecture,
- Lecture du code source pour trouver le code pertinent (bottom-down),
- Regroupement de sections de code,
- Tranchage ou cisaillement de code en section/module,
- Identification de balises (c.-à-d. des énoncés repères dans le code).

Il n'existe pas un seul modèle cognitif englobant pour la lecture du code source. Un développeur va changer de stratégies régulièrement et même en utiliser plus qu'une à la fois. La découverte d'une bonne stratégie est considérée comme un art créatif.

La capacité de lire une section de code source et d'être capable de décrire les relations internes et externes de cette section est considérée comme une compétence intermédiaire (Lopez, Whalley, Robbins, & Lister, 2008). De plus, des connaissances du domaine augmentent de façon significative la capacité d'un programmeur de bien comprendre du code source.

Si des outils informatiques sont utilisés dans ce processus, il est important de s'assurer que ceux-ci aident à la réduction du processus cognitif. Cependant, il est fort possible qu'un outil puisse forcer l'utilisation d'une stratégie qui est mal adaptée pour le problème actuel.

#### 2.5.3.10 L'utilisation d'outil pour automatiser la compréhension

Lorsqu'un programmeur fait face à un problème, la première réaction typique est toujours : « on peut écrire ou trouver un utilitaire ou un logiciel pour faire ceci ! ». Effectivement, il est très facile de trouver de multiples outils qui ont été créés pour aider la compréhension du logiciel. Ce genre d'outil peut grandement aider la compréhension d'un logiciel et le recouvrement de l'information de celui-ci (Alnusair & Zhao, 2010).

La majorité de ces outils appartiennent à une de ces trois catégories :

- Analyses statiques,
- Analyses dynamiques,
- Visualisation de code.

Chaque genre d'outil aide un type de stratégie de compréhension utilisé par un programmeur (Kumar N. , 2013). Il est donc naturel d'en utiliser plusieurs pour mieux comprendre l'application. Cependant, certains outils peuvent imposer une stratégie qui n'est pas compatible avec l'exercice en cours, l'application ou même l'usager (Storey, Wong, & Muller, How do program understanding tools affect how programmers understand programs?, 1997). Il faut trouver le bon outil pour ce problème.

Il faut aussi mentionner que les outils sont rarement intégrés avec d'autres outils, et que les échanges et les partages de données sont plus difficiles que désiré (Jin & Cordy, 2006).

#### 2.5.3.11 Analyses statiques

Ce genre d'outil parcourt le code source pour extraire des informations structurelles pour aider comprendre le logiciel. Ces outils utilisent plusieurs approches différentes, telles que :

- le code source pour recréer le design et les relations des classes, les fonctions, les fichiers, etc…,
- Analyse du code source pour déceler des patrons de conception (un exemple est le projet FAMOOS) (Demeyer, Ducasse, & Nierstrasz, 2002),
- l'analyse du processus de construction du logiciel (outils tels que : Make, Nmake, Maven, Ant et les propriétés du projet Visual Studio),
- extraire les commentaires à l'intérieur du code source,
- reconstruire l'historique du code source en parcourant l'historique des soumissions (par exemple à l'aide d'un « commit » ou un « check-in ») et les commentaires de ceux-ci dans un logiciel de gestion de versions pour comprendre les modifications qui ont été faites avec le temps et leurs raisons d'être.

Chaque outil et source d'information peuvent aider dans le processus de compréhension d'un logiciel en fournissant de l'information pertinente.

Mais il est aussi facile pour qu'un outil puisse nuire à sa compréhension. Les outils peuvent présenter des informations non pertinentes ou confuses. Ceci est causé par :

- une architecture de classes incompréhensibles,
- des commentaires incorrects, non pertinents ou trop vieux,
- des soumissions dans un gestionnaire de versions incorrect, illogiques, avec des commentaires erronés,
- L'utilisation d'une particularité ou fonctionnalité d'un langage de programmation
- L'utilisation de pointeurs, polymorphisme, chargement de classe dynamique (Canfora, Di Penta, & Cerulo, 2011).

#### 2.5.3.12 Analyses dynamiques

Plusieurs outils analysent plutôt les journaux applicatifs lors de l'exécution du logiciel. Ceci inclus l'analyse des journaux de système, le débogage applicatif, et l'utilisation d'outils de systèmes, tel que :

- environnement Unix : Truss, Ptrace ou Strace,
- environnement Windows : Windows Performance Toolkit, Xperf et Process Monitor.

Les journaux créés ont tendance à être très volumineux et lourds. De plus, ce genre d'analyse est très sensible à la quantité, qualité et variation des entrées à l'application. Il est important de choisir les aspects précis que l'on désire comprendre et de bien préparer l'exécution de l'application pour créer des journaux qui contiendront les informations recherchées.

#### 2.5.3.13 Visualisation de code

Les outils de visualisation de code sont des outils qui ajoutent une composante graphique et visuelle à l'analyse statique et l'exploration du code. Il existe beaucoup d'outils de ce genre. Certains sont spécialisés dans un langage de programmation, d'autres sont plus génériques.
Des exemples de ce genre d'outils sont présentés. Il est à noter que les deux premiers sont le résultat d'activité académique et sont aussi un peu âgés.

**RIGI**

- Outil interactif pour la compréhension du logiciel et pour recréer la documentation d'un logiciel,
- Créer par le professeur Hausi A. Müller de l'université de Victoria en Colombie-Britannique il y a plus d'une décennie.

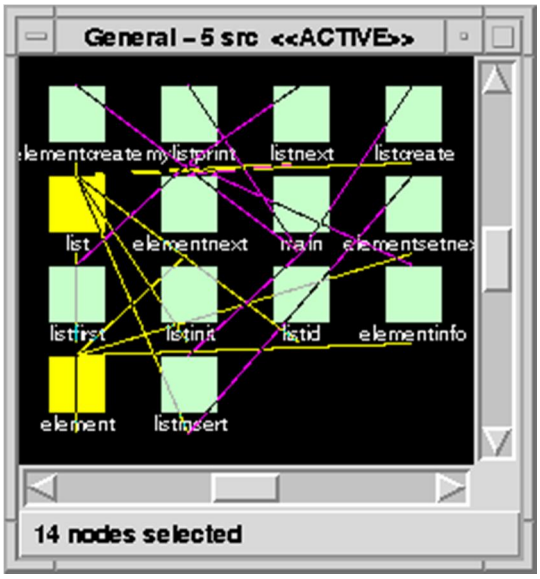


Figure 2 - Outil Rigi

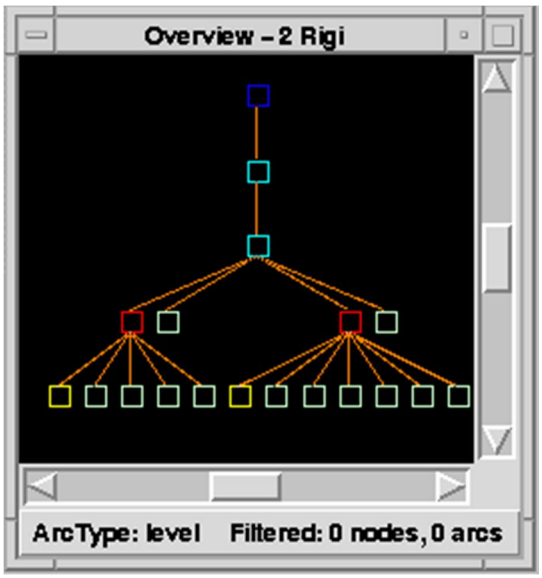


Figure 3 - Outil Rigi

**SHriMP**

- « Simple Hierarchical Multi-Perspective views »,
- Professeur Margaret-Anne D. Storey de l'université de Victoria en Colombie-Britannique.

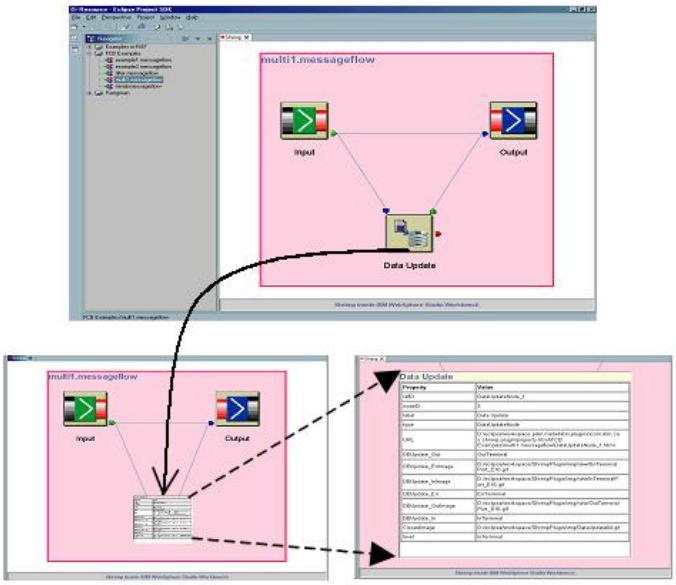


Figure 4 - outil SHRIMP

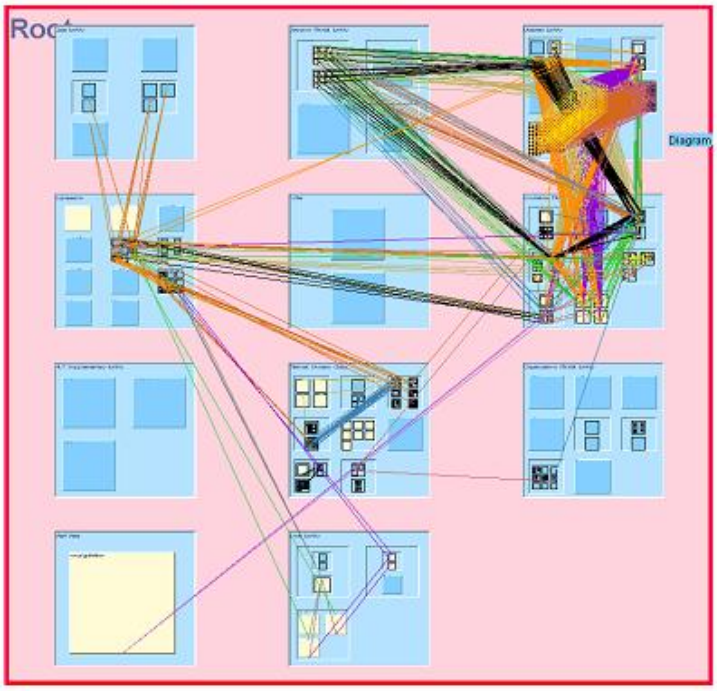


Figure 5 - outil SHRIMP

**Understand**

- http://www.scitools.com/.

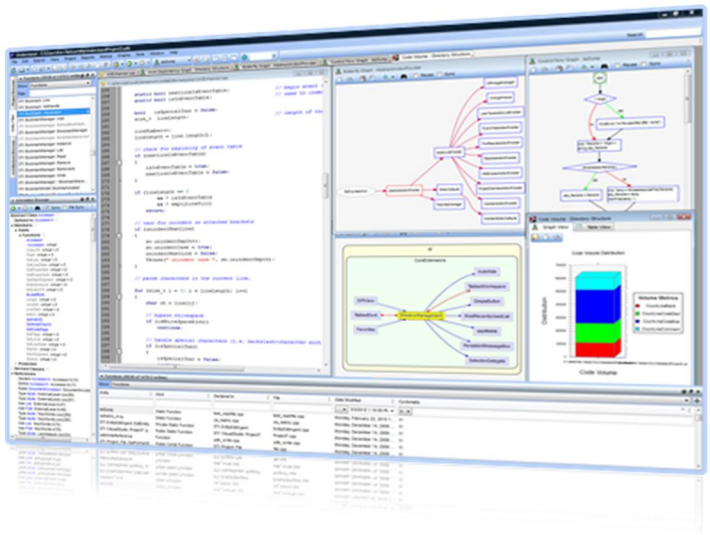

Figure 6 - outil Understand

IBM Rational

Les outils de la famille IBM Rational, incluant « IBM Rational Purifyplus ».

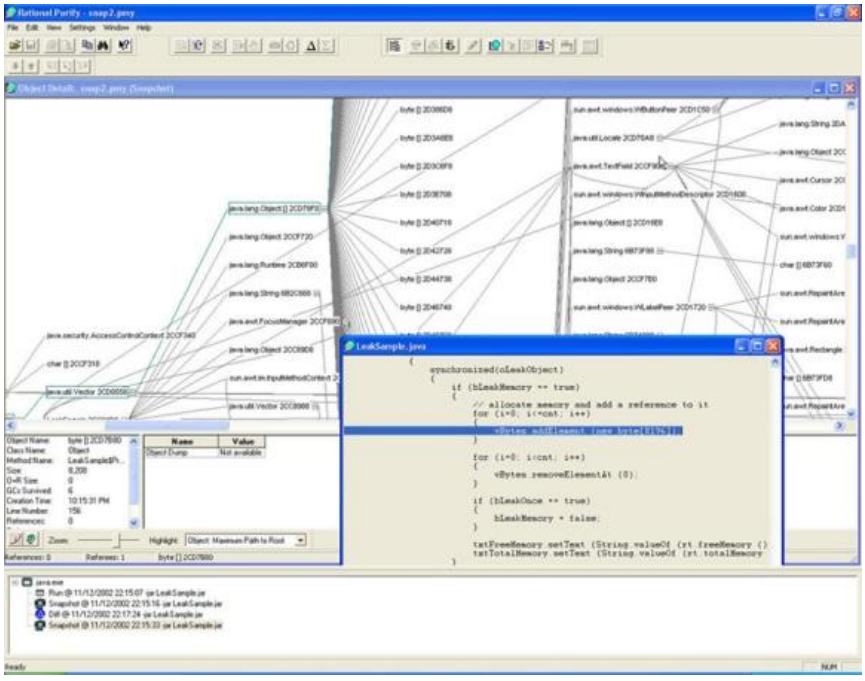

Figure 7 - outil IBM Rational

## 2.6 Sommaire des outils

L'utilisation d'un outil peut aider la compréhension d'un logiciel. Il est typiquement assez rare de trouver un outil qui permet l'utilisation de plus d'une méthode (c.-à-d. statique, dynamique ou visuelle) (Kumar N. , 2013), donc l'utilisation de plusieurs outils différents est souvent nécessaire.

Cependant, l'utilisation de ce genre d'outil en industrie n'est pas assez répandue (Storey, Fracchia, & Müller, Cognitive design elements to support the construction of a mental model during software exploration, 1999).

Lorsqu'il y a utilisation de ce genre d'outil :

- C'est plutôt un effort isolé par un petit groupe d'individus (possiblement singulier), au lieu d'un effort généralisé par une équipe,
- Il y a un effort requis pour l'apprentissage de l'utilisation des outils (courbe d'apprentissage),
- L'apprentissage de l'outil va main dans la main avec la compréhension d'un logiciel, car un programmeur qui suit les tutoriels de l'outil va certainement utiliser le logiciel à comprendre,
- Il est difficile pour un programmeur de partager avec un collège ses nouvelles connaissances techniques de l'outil et sa compréhension améliorée du logiciel. Donc le cycle d'apprentissage est à recommencer.

Les outils peuvent aider de façon significative la compréhension d'un logiciel. Malheureusement, il n'y a pas un outil intégrateur qui peut simplement faire l'analyse et représenter l'architecture tout en recréant les artéfacts d'un logiciel. Ces outils peuvent appuyer le programmeur lors de l'étape de formation d'un modèle cognitif d'un logiciel qu'il ne connait pas.

## 2.7 L'éducation de la maintenance logicielle à l'université

Une étude universitaire démontre que les compétences d'écriture de code des étudiants augmentent de manière marquée lorsqu'il augmente, de seulement 10%, leurs activités de « lecture de code source » (Kumar A. , 2013) (Lopez, Whalley, Robbins, & Lister, 2008). Cette amélioration des compétences de lecture et écriture de code source sera bien utilisée lors des travaux de compréhension d'un logiciel.

Lors de leur premier emploi après l'université, il est fort probable que la majorité des diplômés vont commencer leurs carrières dans la maintenance logicielle, et même possiblement y seront pour toutes leurs carrières. La raison pour ceci est que les nouveaux projets du genre « on recommence à neuf » se font de plus en plus rares en entreprise (Akkiraju, Mitra, & Thulasiram).

Il est suggéré d'augmenter la présence d'exercices de lecture du code source non seulement au niveau universitaire, mais aussi dans l'industrie. Il serait souhaitable d'effectuer de la formation à ce chapitre. De plus, il serait intéressant d'ajouter des techniques d'écriture de code et de préparation à la rétro-ingénierie. L'objectif étant de faciliter la maintenance des logiciels existants (Canfora, Di Penta, & Cerulo, 2011).

## 2.8 Solutions proposées

- Changer la perception de la maintenance et de la rétro-ingénierie pour encourager l'évolution d'un logiciel, et non simplement de corriger le problème de nos prédécesseurs,
- Encourager la lecture de code source pour améliorer les compétences d'écriture de code,
- Encourager l'utilisation des pratiques de génie logiciel moderne.

## 2.9 Rhétorique – les obstacles de la rétrotraçabilité

- La majorité des applications modernes sont rarement conçues de la case zéro,
- Un système/solution/application informatique n'a pas qu'une architecture : elle en a plusieurs,
- La rétro-ingénierie est un processus à plusieurs étapes,
- La maintenance logicielle représente de 60% à 90% des efforts dans le cycle de vie d'un logiciel (Hill, 2008) (Storey, Fracchia, & Müller, Cognitive design elements to support the construction of a mental model during software exploration, 1999),
- La compréhension d'un logiciel représente une portion substantielle dans le processus de maintenance logiciel,
- La compréhension d'un logiciel exige beaucoup de temps,
- La gestion du changement et des variabilités est importante dans la vie d'un logiciel (Acher, et al., 2011).

Il n'y existe pas une solution unique pour comprendre du code source. Chaque solution ou technique utilisé pour comprendre un logiciel ont un mérite, mais peuvent aussi avoir des lacunes. Typiquement, plusieurs solutions/techniques seront utilisées, de façon interchangeable et au besoin.

# CHAPITRE 3

## Revue de la littérature Gestion des processus d'affaires

### 3.1 Introduction

Selon l'approche de planification du cadre de Basili, utilisée pour ce projet, la première étape consiste en une revue de la littérature qui se concentre, pour ce projet, sur le BigData, le NoSQL , et le domaine du BPM.

### 3.2 Étendue des critères de recherche

Lors du processus de recherche, les termes suivants ont été utilisés, en français et anglais:

- BPM – Business Process Management,
- BigData, mégadonnées,
- NoSQL,
- Cloud computing, infonuagique, nuage.

### 3.3 Sommaire / Synthèse des thèmes

- BigData :
    - Utilisez le bon engin pour les bonnes données,
    - L'indigestion de trop de données inutiles,
    - « Garbage in – garbage out ».
- BPM
    - Évolution depuis les années 1990,
    - Beaucoup plus d'intérêt, surtout depuis que SAP offre des processus,
    - multiple standard de notation,
    - Évolution vers le iBPM,

- BPM et BigData
    - Utilisation de BPM pour faire du nettoyage des données dans une base de données BigData.

## 3.4 Méthodologie

Cette section est une revue de la littérature en trois parties :

- Discussions concernant la gestion des processus d'affaires (GPA) (c.-à-d. le BPM),
- Discussions concernant le BigData (c.-à-d. les mégadonnées),
- Discussions sur l'utilisation conjointe de GPA et mégadonnées.

## 3.5 Discussions sur GPA (BPM)

Le domaine de la GPA aussi connu sous le nom « Business Process Management (BPM) » en anglais.

La gestion des processus d'affaires est définie comme étant l'automatisation des processus d'affaires, de manière complète ou partielle, lorsque des documents, informations ou tâches sont passés d'une personne à l'autre pour être traités selon des ensembles de procédures et règlements. (Dumas, La Rosa, Mendling, & Reijers, 2013) (Bergmann, 2007)

La GPA n'est pas exclusivement une solution de technologie, mais un ensemble de responsabilité, pratique et gestion qui inclut l'utilisation de la technologie. La GPA peut être divisée de la façon suivante :

- Les processus, typiquement sous la responsabilité des analystes d'affaire de l'entreprise. Les analystes d'affaires utilisent un langage, outils, des systèmes et des procédures de gestion pour travailler sur les processus
- Les informations, documents (de clients, ou internes) qui entrent dans l'entreprise ou sont créés par celle-ci ou des tâches qui doivent être accomplies

- Les systèmes de GPA du groupe TI (typiquement hétérogène), qui peuvent inclure :
    - Outil de modélisation de processus avec interface graphique,
    - Des engins de traitement qui sont intégrés dans des systèmes existants, tels qu'un système de base de données, ou un système intergiciel aussi connu en anglais "middleware",
    - Journalisation et analyse des activités et suivi des processus.

Le GPA se positionne entre les analystes d'affaire de l'entreprise (c.-à-d. créateur et responsable des processus d'affaires) et les analystes d'affaire du groupe TI (c.-à-d. facilitateur et support des processus d'affaires). Ces deux groupes ont un langage et une façon de travailler qui sont différents un de l'autre, et le GPA offre un pont, quoique faible, entre les deux groupes.

La faiblesse du pont s'explique parce que le travail des analystes d'affaires d'entreprise doit quand même être « converti » ou traduit dans un langage qui peut être utilisé par les systèmes d'informations, tels que les engins de traitement de flux de travail. Il y a eu beaucoup de progrès pour réduire cette étape de traduction, mais aucune littérature trouvée n'a démontré que les outils d'aujourd'hui permettent aux analystes d'affaire d'entreprise d'effectuer des modifications de processus d'affaires dans un outil informatique GPA sans l'assistance des outils ou un membre du groupe TI.

Le GPA souffre aussi d'un problème de perception initial. Comme mentionné précédemment, une solution GPA est typiquement hétérogène. Une de ses composantes est un outil de modélisation typiquement avec une interface graphique, telle que le logiciel de source libre Oryx. (en anglais : « workflow editors ») Il est très facile de faire une présentation avec ce genre d'outil, mais beaucoup plus difficile de démontrer les autres aspects d'une solution GPA. Malgré cette lacune, l'utilisation de solutions GPA est encore en croissance depuis le début des années 90.

En plus de la définition du GPA qui a été présenté au début de ce chapitre, la littérature a offert plusieurs autres définitions, tel que :

- Le GPA représente une sphère d'activité qui a de multiples niveaux qui rejoignent plusieurs points de vue, aspects et objectifs à la même place :
    - Les gestionnaires d'affaires y sont attirés vers le GPA car celui-ci a su démontrer sa capacité d'améliorer la performance organisationnelle, d'aider avec la conformité règlementaire et qualité du service,
    - Les spécialistes TI apprécient le GPA pour sa capacité de leur offrir un langage commun qu'ils peuvent utiliser pour communiquer précisément avec les usagers et les parties prenantes.
- L'utilisation du GPA offre un point de rendez-vous à plusieurs groupes disparates. Ces groupes ont tendance à vivre dans leur propre monde. L'utilisation du GPA offre un langage commun pour bien comprendre et communiquer le fonctionnement interne d'une entreprise,
- L'utilisation du GPA est l'art et la science de comprendre et de gérer la façon dont le travail est accompli dans une entreprise. Il permet de suivre le processus et de s'assurer une consistance des résultats. L'objectif est de tirer avantage des opportunités d'amélioration qui s'offrent à l'entreprise. (Dumas, La Rosa, Mendling, & Reijers, 2013),
- L'objectif du GPA n'est pas d'améliorer l'efficacité d'une activité individuelle, mais plutôt de gérer la série complète d'activités, actions et décisions dans une procédure, toujours avec l'objectif d'ajouter de la valeur à l'organisation et ses clients. (Dumas, La Rosa, Mendling, & Reijers, 2013),
- Un processus est une collection d'évènement, activités et décision qui collectivement mène vers un résultat qui va offrir une valeur aux clients de cette organisation. Toutes les organisations ont des processus. Comprendre et gérer les processus pour s'assurer qu'ils produisent des résultats et des valeurs identiques sont l'ingrédient clé pour l'efficacité et la compétitivité d'une organisation. (Dumas, La Rosa, Mendling, & Reijers, 2013),

- Le GPA est un ensemble de principes, méthodes et outil pour créer, analyser, exécuter et surveiller les processus d'affaires (Ko, 2009).
- Toutes les activités humaines sont une série de processus, que ce soit une planification d'une fête jusqu'à la gestion d'un processus complexe de fabrication. (Ko, 2009).

## 3.6 Définitions

Il y a deux genres de systèmes de gestion des processus : ceux qui gèrent les activités et ceux qui gèrent les entités (documents, etc.). (Bergmann, 2007) La gestion des processus d'affaires et la gestion du flux de travail fournissent les méthodologies et logiciels qui aident à organiser les processus qui fonctionnent à l'intérieur d'une organisation. (Bergmann, 2007)

Dans plusieurs SGPA, la création de journaux et la production de rapport pertinent requièrent encore l'utilisation d'outils externe. (Bergmann, 2007)

La gestion des processus d'affaires (GPA) est une discipline de gestion axée vers les processus. Ce n'est pas une technologie. Les systèmes de la gestion de processus d'affaires (SGPA) sont des technologies qui se retrouvent dans des solutions de GPA et autres catégories de produit informatique. (Ko, 2009)

La modélisation et la gestion des processus d'affaires font partie intégrante de la stratégie d'une entreprise moderne, qui doit absolument améliorer et optimiser de façon continuelle son efficacité opérationnelle. Une étude du groupe Gartner, en 2011, mentionne que l'amélioration des processus d'affaires a été identifiée comme étant une des « attentes des groupes TI » pour les cinq dernières années. (Ami & Sommer, 2007)

L'adoption de la loi SarbanesOxley(SOX) force les entreprises américaines à subir des vérifications de la conformité de façon régulière. Ceci augmente la fréquence et l'importance des vérifications des processus des groupes TI qui sont nécessaires. (Ko, 2009)

L'adoption de la GPA n'est pas assez, mais il faut aussi gérer attentivement la progression des processus. Toyota a déjà mentionné que la définition d'un processus d'affaires ne représente que 10% de valeur ajoutée. Si la progression de celle-ci n'est pas surveillée, il y a potentiellement 90% de perte de la valeur de l'implantation de la GPA. (Ko, 2009)

Les entreprises qui ont la plus grosse chance d'accomplir des projets de GPA avec succès sont celles qui n'ont pas encore atteint un niveau de « politique interne » trop avancé. Typiquement, on retrouve des équipes qui perçoivent des opportunités de croissance dans évidemment des entreprises en croissance. Ceux-ci sont plus réceptifs à adopter le GPA. On retrouve aussi beaucoup plus de résistance à cette adoption du GPA dans des entreprises plus stables, d'où l'attitude de « protection de territoire » et de silos. (Sinha)

Même si une entreprise n'a pas de programme de GPA, les processus d'affaires sont souvent utilisés de façons informelles et font partie de l'expertise, compétence et expérience d'un employé. Mais avec les années, l'entreprise réalisera qu'elle doit définir et décrire ses processus de façon formelle. L'entreprise devra, pour des raisons diverses, les documenter, les comprendre, les communiquer, les automatiser et doit s'assurer de la conformité et de la cohérence et finalement les améliorer. (Monsalve, 2012)

Un des plus grands défis du GPA a toujours été l'homme. La nature du GPA exige que celle-ci traverse les multiples divisions, groupes, entité et base (c.-à-d. en silos) de pouvoir à l'intérieur d'une entreprise. Elle a toujours été perçue comme une menace aux aspirations et enjeux politiques de directeurs ambitieux. (Hill J. , 2013)

Il a été mentionné que deux des plus gros défis pour l'adoption du GPA sont :

- Le changement continuel des conditions du marché,
- Les conflits organisationnels, aussi connus comme « la politique interne ».

Il y a toujours eu un gouffre entre les grandes connaissances des analystes d'affaires et leurs incapacités de transcrire ces connaissances dans un format électronique et utilisable sans l`aide du groupe TI. (Gao, 2013)

Il existe plusieurs méthodologies qui intéressent les adeptes du GPA. Cependant, le concept de base est de créer une représentation graphique des processus d'affaires d'une entreprise et ensuite les transcrire dans un système d'information de l'entreprise, tel qu'un système PGI « connue en anglais comme ERP », pour que les processus soient facilement gérés et analysés. (Ami & Sommer, 2007)

Le GPA est reconnu comme une approche de gestion qui encourage l'efficacité de l'entreprise en appuyant l'innovation, la flexibilité et l'intégration avec la technologie. (Gao, 2013)

Il existe plusieurs standards de modélisation de processus d'affaires, tel que (Monsalve, 2012):

- « Business Process Model and Notation (BPMN) »
- « Event-Driven Process Chains (EPC) »
- « Integrated DEFinition methods (IDEF) »
- « Petri Nets »
- « Role Activity Diagrams (RAD) »
- « Unified Modeling Notation (UML) »
- « Yet Another Workflow Language (YAWL) »
- et bien sûr le langage Qualigramme

Plusieurs auteurs mentionnent les problèmes vécus pour trouver une notation de modélisation qui permet la communication et la participation de toutes les parties prenantes. (Monsalve, 2012) D'autres auteurs mentionnent que les certaines notations GPA sont hautement complexes pour tenter de satisfaire les différentes perspectives de toutes les parties prenantes. L'exemple qui est donné est celui-ci : un modèle de documentation pourrait paraître différent d'un modèle qui veut démontrer l'automatisation du même processus. (Monsalve, 2012)

Si tous les processus d'affaire d'une organisation sont optimisés, l'organisation devrait être plus efficace et productive pour atteindre son objectif de satisfaire ses clients, et ultimement, d'améliorer ses profits. (Monsalve, 2012) La modélisation des processus d'affaires est l'étape de produire des représentations et modèles abstraits des processus d'affaire en place. (Monsalve, 2012)

Plusieurs auteurs mentionnent qu'il y a quelques années, il y avait plus de 10 groupes formels qui travaillaient sur des standards de GPA. Sept d'entre eux étaient dédiés sur les notations de modélisation. (Ko, 2009) Dans les deux dernières décennies, il y a eu une série de langages de modélisation, standards et logiciels, ce qui a empêché l'adoption du GPA à cause de la confusion et des obstacles présentés lorsqu'il y a trop de choix. (Ko, 2009)

Un problème bien connu est le faible niveau de compétence en modélisation que l'on peut retrouver dans un projet de documentation de processus. (Mendling, Reijers, & van der Aalst, 2010) Malgré l'aide fourni par les outils de modélisation, l'usager typique ne reçoit guère de support dans la création de modèles de processus qui peuvent ensuite être utilisés par un professionnel. (Mendling, Reijers, & van der Aalst, 2010)

Un problème majeur dans ce genre de projet est la quantité énorme de modèles, qui peut facilement être au-delà des milliers et le bas niveau de compétence du modeleur occasionnel. (Mendling, Reijers, & van der Aalst, 2010)

Il existe plusieurs outils de modélisation et il est presque impossible de comprendre les différences fonctionnelles de chacun sans nécessairement les essayer tous. (Yan, Reijers, & Dijkman, 2010)

Bill Gates a dit ceci : « La première loi de l'utilisation de n'importe quelle technologie dans une entreprise est que l'automatisation qui est appliquée sur un processus déjà efficace va

augmenter l'efficacité de celle-ci. La deuxième loi est que l'automatisation d'un processus inefficace va malheureusement amplifier son inefficacité ». (Dumas, La Rosa, Mendling, & Reijers, 2013)

## 3.7 Prochaine évolution : le GPA intelligent (iBPM)

Le groupe Gartner a déclaré que le GPA va évoluer pour inclure les aspects sociaux de l'internet. Il a nommé cette évolution « intelligent BPMS », que nous nommerons le GPAi pour « gestion des processus d'affaires intelligents ».

Le GPAi ajoute un nouveau niveau de fonctionnalité avec l'intégration de technologies d'analyse dans des processus orchestrés. (Gao, 2013)

Les fonctionnalités d'une solution GPAi incluent toutes les fonctionnalités d'une solution typique GPA. Une solution GPAi ajoute des technologies plus évoluées selon le guide de fonctionnalité du groupe Gartner. (Gao, 2013) De plus, le GPAi ajoute les 4 points suivants :

- Analyse : L'une des fonctionnalités les plus intéressantes du GPAi est la capacité d'analyse plus évoluée (Gao, 2013)
- Automatisation : Pour déceler des patrons, identifiée des anomalies et pour extraire des connaissances, la quantité énorme de données requiert des techniques et procédures automatisés, soit complets ou semi-automatiques (Gao, 2013)
- Adaptative : les entreprises sont constamment bombardées de changement, et ses processus doivent suivre la cadence. Ceci demande que la quantité énorme de données qui entrent dans l'entreprise requière une capacité de modifier les critères de capture de façon rapide (Gao, 2013)
- Agilité : Les analystes d'affaires ont une grande connaissance des processus, mais ils ont toujours besoin des spécialistes du groupe TI pour modéliser ces processus dans des solutions GPA. Ce besoin doit être réduit pour permettre un changement par un

> analyste, pour ensuite faire la transition vers un système en production GPA le plus rapidement possible. (Gao, 2013)

Le groupe Gartner positionne treize fournisseurs de solutions qui peuvent fournir l'agilité, l'optimisation des processus et autres fonctionnalités en temps réels qui sont typiquement retrouvés dans la gestion des opérations intelligentes, soutenu par le GPAi (voir section plus bas). Gartner décrit aussi que les entreprises-chefs de file utilisent déjà le GPAi et suggère fortement que toutes les organisations devraient inclure les notions de GPAi dans la planification de leurs architectures GPA. (Sinur, Schulte, Hill, & Jones, 2012)

Gartner a ainsi observé que les entreprises font des efforts pour rendre leurs processus d'affaires plus intelligents. Ils accomplissent ceci avec l'intégration de l'analytique dans les processus et les applications qui supportent ceux-ci. Gartner a nommé cet effort « intelligent business opérations (IBO) », ou les « opérations d'affaires intelligentes (OAI) ».

Le groupe Gartner croit aussi que cette tendance offre aux solutions GPA une possibilité d'évolution de sa capacité et solution offerte. De nouvelles fonctionnalités doivent être ajoutées aux solutions GPA pour les faire évoluer vers une nouvelle génération de solution GPA, pour permettre les OAI. Le groupe Gartner offre cette définition d'une solution GPA :

- une solution GPA traditionnelle offre les technologies nécessaires pour gérer l'interaction entre toutes les ressources d'une entreprise qui contribuent au succès des processus d'affaires, tel que : (Sinur, Schulte, Hill, & Jones, 2012)
    - Les gens, employés,
    - Les systèmes logiciels,
    - L'information,
    - Les règles d'affaires,
    - Les politiques de l'entreprise.

Les gestionnaires d'affaires et les professionnels des TI utilisent les modèles pour permettre une meilleure coopération lors des efforts d'amélioration des processus d'affaires. (Sinur, Schulte, Hill, & Jones, 2012) Le iGPA continue avec toutes les fonctionnalités d'une solution GPA. Une solution iGPA ajoute des fonctionnalités avancés, tel que (Sinur, Schulte, Hill, & Jones, 2012) :

- Les analytiques d'affaires en temps réel,
- Le traitement d'évènement complexe,
- L'intégration des médias sociaux pour permettre l'interaction du comportement social et la collaboration,
- La technologie avancée pour combler les exigences grandissantes de la mobilité.

L'objectif de cette nouvelle fonctionnalité est d'augmenter la capacité d'automatisation et le niveau d'intelligence collectif (c.-à-d. machine et homme). Il a été déjà mentionné qu'une solution GPA a tendance à casser les silos, centre de pouvoir, politique interne d'une entreprise. L'évolution du GPA/GPAi va amplifier ce phénomène et a le potentiel de transformer les environnements de travail traditionnel basé sur des rôles vers un nouveau modèle plus évolué. (Sinur, Schulte, Hill, & Jones, 2012).

# CHAPITRE 4

## Revue de la littérature BigData

### 4.1 Définition et description du problème

Le domaine du BigData, aussi connu sous les noms « mégadonnées » ou « données massives » semble éviter un consensus sur sa définition actuellement. En effet, la majorité des définitions qui ont été retrouvées dans la littérature semblent décrire le problème à résoudre, au lieu offrir une définition du terme. Quelques-unes des définitions ou descriptions du problème sont :

- Des ensembles de données tellement massifs qu'ils sont impossibles ou très difficiles à manipuler avec les systèmes existants de base de données (Fishleigh, 2014),
- Des ensembles de données qui ne peuvent être identifiés, acquis, gérer et traités par des systèmes informatiques traditionnels, outils logiciels et équipements informatiques dans un délai acceptable (Chen, Mao, Zhang, & Leung, 2014),
- Les mégadonnées tentent de résoudre le problème du manque d'extensibilité horizontale de la base de données relationnelle. Le nombre grandissant de requêtes par seconde finissent par faire apparaître des problèmes reliés à la taille des index des tables (LAROCHELLE, 2011),
- Mégadonnées est un terme qui est utilisé pour décrire un volume de donnée trop gros pour être traité par un système de gestion de base de données standard, et donc requiert une solution alternative (Lauzon, 2012),
- Les « mégadonnées » dont des ensembles de données qui dépassent la capacité des systèmes SGBD classique. Les données sont trop volumineuses, ont besoin de transiter de façon trop rapide ou ne rencontrent pas la structure de l'architecture de la base de données. (Lauzon, 2012)

Toutes ces définitions semblent indiquer un aspect d'une date sans préciser celle-ci. Seront-elles plus précises dans cinq années? Une définition plus intéressante a été avancée par Matt Benati de la société Attunity (Fishleigh, 2014) :

> « L'analyse et les résultats qui sont possibles lorsque vous utilisez une puissance de computation énorme avec un ensemble de données absolument massif. »

## 4.2 Les outils BigData pour ce projet

La solution technologie de BigData qui nous intéresse plus particulièrement provient du projet de logiciel libre Hadoop de la fondation Apache. Ce projet est basé sur l'architecture technologique de Google, qui a commencé à dévoiler publiquement celle-ci dès 2004. Le système de fichier distribué Hadoop (c.-à-d. le Hadoop Distributed File System (HDFS)) est un système de fichier distribué utile pour des applications qui nécessite le traitement d'une quantité massive de données, tout en fonctionnant sur des ordinateurs de commodité (c.-à-d. des ordinateurs à bas prix). (Cryans, April, & Abran, 2008)

Une solution dite de BigData n'utilise pas un seul logiciel. C'est plutôt un ensemble de logiciels, réseaux et serveurs qui fonctionnent ensemble pour offrir, facilement, une solution puissante de parallélisation des tâches. Il y a eu plusieurs mentions précisant que le BigData est un écosystème (Sawant & Shah, 2013). L'écosystème Hadoop est composé de plusieurs logiciels compatibles entre eux (par exemple : Hive, Pig, HDFS, Ambari, et bien d'autres…), chacun ayant une fonctionnalité particulière et nécessaire pour l'écosystème.

Une solution BigData nécessite aussi l'utilisation d'un système de gestion de base de données (un SGBD) d'un nouveau type que l'on nomme le « NoSQL ». Il existe plusieurs systèmes de base de données « NoSQL ». Chacune d'elle possède ses forces et ses faiblesses afin de résoudre un problème particulier, tel que (Lauzon, 2012) :

- MangoDB : plus adaptée pour le stockage de documents,

- HBase : qui fonctionne très bien avec des données qui doivent être identifiées par une clé unique,
- Impala qui offre un langage semblable au SQL  et qui est d'une rapidité incroyable pour la mise à jour des données.

### 4.3 Pourquoi utiliser les « mégadonnées »

Il est évident que les entreprises sont de plus en plus impactées par la croissance des données obtenues et qui sont créées par celle-ci. Et cette tendance ne va qu'en augmentant. Il est aussi très difficile pour eux de comprendre et interpréter ces données (Faura, 2012).

Comme discuté, les entreprises font face à des problèmes lorsque le volume de donnée augmente sans cesse, et que le délai requis pour faire une requête devient de plus en plus pénible et créent des délais de plus en plus longs. Les coûts d'achat d'augmentation de la capacité de calcul (c.-à-d. l'ajout constant de serveurs) et les coûts de mise à jour des licences du SGBD deviennent de plus en plus importants (par exemple les licences d'Oracle).

Afin d'améliorer la situation et la performance d'un système SGBD existant, les spécialistes ont recours à des techniques telles que : cesser d'utiliser certaines fonctionnalités du SGBD. Par exemple on peut enlever des index ou des propriétés ACID (ce qui signifie A:atomicité, C:cohérence, I:isolation et D:durabilité), où même procéder à la dénormalisation des données. Toutes ces optimisations réduisent la flexibilité et le rend de plus en plus difficile la maintenance du logiciel. Ces approches forcent celui-ci à fonctionner d'une manière dont il n'a pas été conçu initialement et crée toutes sortes d'effets pervers.  (Lauzon, 2012)

Une solution actuellement populaire, à ces problèmes, est de changer de technologie et d'abandonner les bases de données relationnelles pour aller vers les bases de données NoSQL.

## 4.4 Pourquoi ne pas utiliser BigData

J'ai été aussi surpris de trouver certains auteurs (Stucchio, 2013) (Woodie, 2014) qui avisent leurs lecteurs de penser sérieusement aux justifications avant d'entreprendre l'utilisation des technologies du BigData. Ces auteurs posent la question suivante: « avez-vous vraiment une grande quantité de données? ». Ils tentent ainsi d'éveiller le lecteur à la possibilité qu'il y ait d'autres solutions et technologies possibles avant de passer au BigData.

Sur un site web du genre question/réponse, un usager a posé une question pour l'apprentissage de BigData. Il demande s'il est possible de trouver du matériel d'apprentissage qui peut s'exécuter sur son ordinateur portatif. Un contributeur lui répond qu'il n'y a pas vraiment une démarcation précise entre la notion de BigData et « une quantité de données massive ». (Spener, s.d.) Ce contributeur mentionne qu'il est très facile de manipuler des millions de données avec le langage Python et les librairies Pandas (c.-à-d. Python Data Analysis Library, s.d.). Donc pourquoi aller avec un écosystème de BigData.

Ce contributeur a, de toutes évidences, un penchant personnel pour le langage Python, mais sa réponse démontre que les technologies des écosystèmes du BigData ne sont pas les seules solutions technologiques pour l'analyse de grandes quantités de données. Cependant, ma lecture me fait croire que si le nombre de données est plus proche des milliards, l'utilisation d'un écosystème BigData est une solution éprouvée et de plus en plus recommandée.

Le message est clair : il ne faut pas se laisser emporter par la vague de popularité des solutions BigData aveuglément. Chaque problème ayant plusieurs solutions possibles, il s'agit de trouver celle qui convient le mieux selon chaque cas. Ainsi il faut bien évaluer si l'utilisation d'un écosystème BigData est souhaitable ou non dans votre cas.

## 4.5 Le succès avec les écosystèmes BigData

Plusieurs entreprises ont démontré que l'utilisation correcte de l'écosystème BigData leur a permis de prendre de grandes longueurs d'avance sur leurs compétiteur (Manyika, et al., 2011). Les forces du marché forceront ces compétiteurs à suivre le pas rapidement. Il est estimé qu'une entreprise de distribution ou de revente qui utilise l'écosystème BigData judicieusement peut augmenter son profit d'opération de plus de 60%. Ses compétiteurs ne pourront pas se permettre de rester immobiles s'ils veulent compétitionner.

On estime que le BigData va aussi aider à créer de nouvelles opportunités de croissance et même de nouvelles catégories d'entreprise, telle que celles qui accumulent, agrègent et analysent des données et les revendent.

## 4.6 La quantité et la qualité des données. Sont-elles compatibles?

Il n'a pas été possible de trouver une publication qui décrit une limitation concernant la quantité de données qui peuvent être traitées, mais plusieurs documents, discutant de la problématique de la qualité des données. Une fois qu'un système BigData est mis en place, la littérature discute abondement de deux problèmes majeurs qui surviennent avec la collecte et l'emmagasinage de quantités massives de données :

- La qualité des données (Sawant & Shah, 2013),
- La quantité de données est si massive qu'il est possible que ceci empêche l'être humain d'absorber, interpréter et agir sur ceux-ci.

### 4.6.1 La qualité des données

Plusieurs auteurs décrivent l'échec des approches impliquant des mégadonnées. Même si aucun auteur ne mentionne la notion qu'il est possible d'avoir trop de données, plusieurs sont

unanimes pour décrier la qualité des données qui sont insérées dans un système de mégadonnées. On peut lire des commentaires tels que;

- Il n'y a pas de surcharge de donnée, mais plutôt un échec dans le filtrage des données entrantes,
- Un des plus gros défis pour bien comprendre et prioriser les données est de trier les vraies données et éliminer les données que l'on peut considérer comme ordures,
- Il y a probabilité que 90% des données entrantes peuvent être considéré comme du bruit,
- Il faut séparer le « bruit » des informations pertinentes. Le ratio de bonne donnée est de 10:90.

Toutes les sources de données doivent être gérées de façon organisée pour qu'une entreprise soit capable d'extraire de l'information pertinente. Lorsqu'une entreprise adopte les « mégadonnées », elle doit se doter de nouveaux outils et expertises de gestion de donnée, stockage et analyse de données pour en tirer profit.

### 4.6.2 Agir correctement sur cette quantité de données

Il est important de bien comprendre que l'objectif de l'utilisation des mégadonnées est de prendre un ensemble de données possiblement non structuré et de créer de l'information décisionnelle, suivi d'une action (Fishleigh, 2014). Cependant, il est possible que l'être humain ne soit pas capable d'absorber, interpréter et agir sur ce volume de données (Manyika, et al., 2011).

Cependant, même si les données dans un système mégadonnées sont propres, correctes, pertinentes et précises, il faut aussi avoir les connaissances, compétences et outils pour faire l'analyse de ces données pour créer une action. Si vous avez simplement l'accès aux données et de beaux graphiques, ce n'est pas une garantie que vous serez capable d'augmenter les profits et la compétitivité de votre entreprise.

Les individus qui font ce genre d'analyse doivent avoir un mélange de talent pour être capables de faire l'extraction intelligente de données, aussi connue comme « les sciences des données ». Un auteur offre cette description: « une personne qui excelle à la manipulation et à l'analyse de données, surtout les ensembles de données massives qui ne peuvent pas être utilisés dans un tableau d'un chiffrier » (Fishleigh, 2014).

Les talents désirés pour accomplir l'extraction de connaissances sont :

- Analyse de statistiques,
- Exploration de données,
- Connaissances des processus d'affaires,
- Capacité de communication,
- Compétence avec la technologie et logiciel.

Une entreprise a besoin d'un processus qui va l'aider à activer les données d'une façon intuitive et facilement. Elle doit permettre à ces individus de bien accomplir leurs travaux. De plus, plusieurs auteurs mentionnent qu'il est préférable de découvrir une réponse approximative aux bonnes questions, que d'avoir la réponse précise à une mauvaise question.

## 4.7 Les composantes requises pour traiter les mégadonnées

Un auteur (Sawant & Shah, 2013)souligne qu'une solution mégadonnées doit répondre aux exigences de ce genre d'environnement :

- Roulement/insertion/ des données,
- Le type, genre et variété des données (structuré et non structuré),
- Complexité des données,

## 4.8 Et évidemment, la quantité ou le volume de données

Il y a trois composants qui sont critiques pour comprendre les besoins d'un projet et d'une solution (Fishleigh, 2014) mégadonnées :

- L'importance de la rapidité de transfert et la qualité des données entrantes dans un système impliquant des mégadonnées,
- De bien interpréter la quantité énorme d'éléments de données structurés et non structurés et de créer des graphiques et outils qui sont utilisables dans une analyse plus poussée,
- La fouille de donnée dirigée par la sémantique.

## 4.9 L'architecture Big Data

L'architecture d'un système impliquant des mégadonnées est typiquement hétérogène, aussi connue comme étant un écosystème. Il est possible de trouver plusieurs solutions différentes pour résoudre le même problème. Un auteur mentionne qu'il est critique de bien choisir l'architecture de cette solution, car celle-ci permet d'exploiter la puissance des mégadonnées (Sawant & Shah, 2013).

# CHAPITRE 5

## Revue de la littérature – Il faut mentionner le « cloud »

### 5.1 Infonuagique, le nuage, le « cloud »

Dans les sections précédentes, ce document a fait très peu de mentions du … «cloud ou nuage». Cependant, il est impossible d'ignorer l'aspect infonuagique pour n'importe quelle solution informatique (Hill J. , 2013).

Il existe plusieurs autres noms pour cette architecture, tel que :

- Le « cloud computing »
- l'informatique en nuage
- l'informatique en nuagique
- le Cloud

L'infonuagique est le terme proposé par l'Office québécois de la langue française.

> INFONUAGIQUE
> MODÈLE INFORMATIQUE QUI, PAR L'ENTREMISE DE SERVEURS DISTANTS INTERCONNECTÉS PAR INTERNET, PERMET UN ACCÈS RÉSEAU, À LA DEMANDE, À UN BASSIN PARTAGÉ DE RESSOURCES INFORMATIQUES CONFIGURABLES, EXTERNALISÉES ET NON LOCALISABLES, QUI SONT PROPOSÉES SOUS FORME DE SERVICES, ÉVOLUTIFS, ADAPTABLES DYNAMIQUEMENT ET FACTURÉS À L'UTILISATION. (Office québécois de la langue française, 2015)

Il ne faut pas être ébloui par les discussions, parades, fanfares et évangéliste au sujet du modèle infonuagique. Il ne faut pas établir un objectif de migration de tous les processus d'affaires d'une entreprise vers le « nuage ». Le groupe Gartner suggère d'utiliser le modèle infonuage pour optimiser les objectifs d'affaires de l'entreprise avec une plus grande flexibilité. (Hill J. , 2013)

## 5.2 Infonuage : menace au directeur de système d'information?

Plusieurs directeurs de système d'information de grande entreprise risquent de perdre le monopole sur le traitement de l'information, car le modèle infonuagique offre la possibilité à un chef de division et de son équipe informatique de simplement utiliser une carte de crédit pour avoir accès à des ressources informatiques plus vaste que celui du groupe TI de l'entreprise. Un directeur de système d'information doit éviter l'érosion de sa sphère d'influence. Ceci est important, car il doit s'assurer de la confidentialité des données d'entreprise. Il doit ajouter à son offre de service les possibilités du BPM et BigData en utilisant les ressources d'entreprise et infonuagiques.

## 5.3 L'infonuage…ce n'est qu'un serveur de quelqu'un d'autre…

Lorsque l'on discute du modèle infonuagique, il ne faut pas oublier que celui-ci peut être simplement défini comme l'utilisation de serveurs d'autrui en utilisant une connexion Internet


Figure 8- L'Infonuagique n'existe pas...vous utilisez l'ordinateur d'autrui (twitter.com, 2015)

## 5.4 Exemple vécu

Le sujet de la sécurité et de la confidentialité des données est un sujet vaste qui est hors contexte de ce document, mais l'auteur offre l'exemple vécu suivant. Deux entreprises internationales avec plusieurs sites dans plusieurs continents et pays fusionnent. Il y a soudainement une panoplie de projets d'intégration entre les deux entreprises, incluant au niveau réseau informatique, équipe de développement logiciel, etc.

Un de ces projets est l'intégration et fusionnement des gestionnaires de code source. Ce projet est divisé en deux volets :

1. migration vers le logiciel SubVersion si une division ne l'utilise pas déjà,
2. migration vers un serveur commun et unique.

Le deuxième volet est problématique, car les deux entreprises se retrouvent avec des sous-réseaux avec des numérotations identiques, et le nouveau groupe fusionner d'architecture réseau n'a pas encore trouvé une solution élégante pour intégrer les réseaux des 2 entreprises.

Donc il est encore impossible de trouver un serveur qui est accessible par tous les sites des deux entreprises en utilisant le réseau interne de celui-ci. Un vice-président suggère avec enthousiasme d'utiliser le modèle infonuagique. Un environnement SubVersion est installé en moins de 24 heures. Un courriel est envoyé vers tous les groupes de développement logiciel : « migrer votre code source vers notre nouvel environnement de gestion du code source nuagique ». Sauf que les problèmes suivants ont été identifiés dans les semaines suivantes:

- certains groupes utilisaient l'authentification avec des groupes dans l'annuaire d'entreprise Active Directory – LDAP. Le nouveau système nuagique ne peut pas faire des requêtes sur cet annuaire,
- certains groupes avaient aussi une liste de distribution courriel associée aux groupes AD-LDAP,
- SubVersion permet l'utilisation de procédures automatiques, aussi connue en anglais de «hook», tel que «pre-commit-hook», «post-commit-hook». Des actions automatiques typiques étaient :
    - Vérification qu'il existe un commentaire d'au moins 10 caractères,
    - Vérification que le commentaire inclus l'identification de la demande de changement (c.-à-d. Mantis – logiciel gestion des bogues),
    - Produire un rapport des différences entre le code source existant et celui qui vient d'être ajouté/modifié,
    - Envoyer un courriel à l'équipe pour aviser du changement.
- Manque de sécurité, car le nouveau système était accessible à tous via l'internet, donc un seul niveau de sécurité… l'authentification simple,
- certains clients ont ajouté des clauses contractuelles pour :
- Interdire le stockage hors du pays du code source utilisé pour des composantes personnalisées,
- interdire l'utilisation et l'accès à ce code source par des non-citoyens du pays en question.

Résultats immédiats :

- un nouvel exercice de réseautique a été nécessaire pour assurer que seulement les employés de l'entreprise avaient accès au système, et ce seulement d'un bureau de l'entreprise. Il a été donc nécessaire d'établir des règles de pare-feu pour permettre seulement l'accès des adresses IP publiques connues de chaque bureau,
- De plus, certains serveurs/gestionnaires de code source local ont été conservés pour respecter les clauses contractuelles de client. Donc un système de gestion de code source à multiniveaux et sites est nécessaire. C'est un doublement d'architecture, ressources et gestion.

Les autres points sont encore en suspens. Un effet qui a été remarqué immédiatement est qu'un niveau d'automatisme et de fonctionnalité a été perdu, et les coûts d'administration de système ont augmenté, même si les ressources sont en Inde.

## 5.5 Leçon apprise

La leçon à retenir est que le modèle infonuagique peut aider une entreprise, mais celle-ci doit gérer son architecture étendue avec autant d'assiduité que si elle utilise ses propres serveurs.

# CHAPITRE 6

## Revue de la littérature – Utilisation de BPM et BigData ensemble

### 6.1 Interaction entre BPM et Big Data

Dans la section GPA, il a été mentionné qu'un des objectifs (c.-à-d. un de nos défis) du GPA est de traverser les multiples divisions, groupes, entité et base (c.-à-d. les silos organisationnels) de pouvoir à l'intérieur d'une entreprise. Les mégadonnées font aussi la même chose pour être capables de fournir un aperçu encore plus global et cohésif de l'information d'une entreprise. L'auteur Clay Richardson du groupe de recherche Forrester croit que le GPA et les mégadonnées donnent l'opportunité à une entreprise de se transformer, en utilisant ce qu'il nomme «big process», l'utilisation du GPA pour activer les mégadonnées, possiblement avec l'utilisation Infonuagique (nuage/cloud) (Davies, 2013).

Plusieurs auteurs sont d'accord avec ce principe :

- « les mégadonnées sont inutiles sans le GPA. » L'utilisation du «big process» pousse encore plus le GPA à travers une entreprise pour fournir des solutions et des améliorations des processus d'affaires à un niveau global dans l'entreprise,
- Un auteur offre cette anecdote : « l'utilisation des mégadonnées sans d'un système de gestion de processus efficace est la même chose qu'avoir une Ferrari sans moteur » (Patnaik, 2013),
- Les mégadonnées et le GPA sont des solutions et produits distincts. En réalité, il n'y a pas d'intersection entre les deux. Mais les deux ont le potentiel d'être une grande influence sur l'autre, et ils ont besoin d'un l'autre (Schooff, 2013)
- Un rapport récent de l'entreprise « Software AG » suggère qu'il ne faut pas se faire distraire avec toutes les discussions au sujet du "nuage", "BigData", les applications mobiles et l'utilisation du côté social de l'internet par les entreprises. Plutôt, il est

possible qu'un investissement dans le GPA puisse être plus profitable à court terme pour plusieurs entreprises (Weisinger, 2013)

- Le défi est de créer une action après l'analyse des données. Une entreprise doit être capable de déclencher rapidement des processus d'affaires pour accomplir les décisions qui seront prises après l'analyse des données. C'est une opportunité pour le BPA. (Davies, 2013)

Un auteur suggère que l'utilisation d'une solution GPA avec des solutions « intelligence d'affaires » et les outils/techniques offerte par les mégadonnées offre une puissance supplémentaire pour les solutions d'analytiques du web, surtout avec de gros volumes de données. Il suggère de plus qu'il est important d'utiliser un niveau d'abstraction avec les processus d'affaires et l'utilisation de l'automatisation pour que les entreprises s'élèvent au-delà des efforts plutôt techniques et les duplications à l'intérieur des processus d'affaires.

Typiquement, l'avantage qui peut être quantifié lors de l'utilisation d'une solution GPA est les économies dans la main-d'œuvre et l'accélération (c.-à-d. la vitesse d'exécution) des processus d'affaires. (Faura, 2012)

# CHAPITRE 7

## Section 2 - Apprentissage de la technologie BigData

### 7.1 Introduction – cadre de Basili

Dans cette deuxième partie du rapport, l'objectif est l'apprentissage de la technologie BigData. Voir le cadre de Basili à l'annexe III.

Les activités à accomplir sont :

- l'installation et configuration d'une BD BigData et le développement d'une (ou plusieurs) application(s) démonstrateur(s),
- Suivre un tutoriel approprié.

Le résultat et la mesure de succès seront l'installation fonctionnelle de HBASE avec logiciel(s) qui démontre(nt) son utilisation.

### 7.2 Vidéos éducationnels sur Youtube.com

Comme première étape, la visite Youtube permet de trouver du matériel éducationnel. Il existe une panoplie de vidéos éducationnels sur Youtube au sujet de Hadoop. J'en ai visionné plusieurs et il y a une quantité impressionnante de vidéo incomplet et comportant des présentateurs non anglophones (c.-à-d. pour les vidéos en anglais) dont on ne comprend pas le contenu. La qualité n'est donc pas toujours au rendez-vous. J'ai par contre apprécié les vidéos suivants :

- Hadoop MapReduce Fundamentals – Lynn Langit (Langit, s.d.)
    - Elle présente une série de 5 vidéos d'une longueur approximative de 50 minutes par vidéo,
    - Il est évident que l'anglais est sa langue maternelle, et sa présentation est de très bonne qualité,

- Elle présente avec beaucoup d'exemple et démonstration, incluant l'utilisation de Microsoft Azure et Amazon AWS,
- Elle présente aussi les problèmes auxquels MapReduce n'est pas destiné à faire face,
- De toute évidence, elle a beaucoup plus d'expertise avec les produits Windows et les interfaces graphiques. Elle démontre aussi l'utilisation de la ligne de commande, mais sa préférence est très visuelle. Ceci nous offre une présentation riche en matière et facile à comprendre,
- Ce que j'ai apprécié le plus, c'est qu'elle a rapidement sauté les exemples typiques de map/reduce (ie : wordcount) et a discuté de sujet avancé tel que :
  - l'utilisation de map/reduce dans la vraie vie,
  - comment optimiser l'exécution incluant l'impact de l'utilisation/genre de la compression,
  - tests unitaires.

- Introduction to NoSQL – Martin Fowler (Fowler, s.d.)
  - Martin est un auteur et un conférencier que j'ai suivi depuis plusieurs années. J'ai acheté plusieurs de ses livres,
  - Il donne une présentation de 50 minutes au sujet de NoSQL,
  - Fait divers : l'origine du terme NoSQL est en réalité un mot-clé utilisé sur Twitter par un groupe d'informaticiens relié à plusieurs projets du même genre pour leur première rencontre/conférence à la fin des années 2000,
  - Il a discuté des différences entre les genres de systèmes Big Data, tel que :
    - Key-value (Project Voldemort, Riak, Redis),
    - Document (mongoDB, CouchDB, Raven DB),
    - Column Family (HBase, Cassandra),
    - Graph (Noe4j).

- Il est intéressant de noter qu'il a expliqué la différence entre les genres de systèmes BigData, mais aussi qu'ils ne sont pas si différents, surtout avec l'utilisation de métadonnées, l'agrégation, etc.
- En effet, son explication de l'utilisation de l'agrégation est très intéressante et serait un sujet d'étude pour le futur. Il spécifie que les systèmes «key-value», «document» et «Column Family» sont du genre agrégation,
- cependant, son explication sur une base de données du genre «Graph» et ses forces pour décrire des relations, me suggère fortement que le choix d'un ou plusieurs des systèmes BigData est très dépendant du problème à résoudre. Un autre sujet d'étude pour le futur,
- Il discute aussi de la consistance des données, et qu'il faut être conscient des différences du traitement des conflits d'écriture. Point pour réflexion : si nos données sont répliquées trois fois, est-ce qu'une action d'écriture garantit que les 3 versions sont à date en même temps, surtout si les systèmes/données répliqués ne sont pas localisés au même endroit? Quel est l'impact si nous avons deux modifications sur deux sites différents? Quelle version est la bonne? Un autre sujet d'étude pour le futur.

Je suis heureux d'avoir choisi de visionner les vidéos de ces deux auteurs. Je les ai trouvées éducationnelles. C'était une excellente première étape.

## 7.3 Tutoriel #1 - Première installation d'un environnement complet - Cloudera

### 7.3.1 Cloudera Express

Pour la première installation, j'ai utilisé l'environnement nommé « Cloudera Express » offert par l'entreprise Cloudera. (Cloudera, s.d.)

### 7.3.2 Un petit sommaire de l'entreprise Cloudera et ses produits

L'entreprise Cloudera existe depuis 2008 et utilise le modèle d'affaire de source libre. C'est-à-dire qu'elle ne vend pas de licence logiciel, car elle n'est pas propriétaire de ceux-ci. Elle offre plutôt une gamme de logiciels de source libres, compilés et bien intégrés ensemble et offerts gratuitement. Elle offre aussi des logiciels de gestions avancées conçues par l'entreprise offerte pour achat optionnelle. Ceux-ci sont empaquetés dans des environnements virtuels qui sont disponibles pour le téléchargement par l'internet et peuvent être utilisés immédiatement. Cloudera offre plusieurs versions de ces environnements :

- «Cloudera express» offre les outils de base de source libre, tels que Hadoop, Sqoop, Flume, HBase, HCatalog, Hive et plusieurs autres. Il n'y a pas de support offert avec cette version. Elle est disponible en mode machine virtuelle qui est téléchargeable, et aussi en mode nuagique. La version «Cloudera Express» est gratuite et peut être utilisée sans limite ni date d'expiration,
- «Cloudera Enterprise» existe en trois versions différentes en utilisant un modèle de souscription annuel. Ce service offre le module «Express», et y ajoute plusieurs outils de gestion, des mises à jour logicielles et une foule d'outils d'intégration dans environnement d'entreprise, tel que l'intégration avec Microsoft Active Directory, les systèmes de répertoire LDAP, le support SNMP, etc.

La version d'entreprise se démarque aussi avec ses différentes options de support offertes directement de Cloudera. Cette version est disponible pour 60 jours à l'essai et nécessite un achat après la période d'essai. Il existe une page web pour décrire les différences d'options entre les deux versions. Celle pour la version courante 5.4 est offerte à ce lien URL : Cloudera Express and Cloudera Enterprise Features :
http://www.cloudera.com/content/www/en-us/documentation/enterprise/latest/topics/cm_ig_feature_differences.html#cmfeig_topic_5_1

De plus, Cloudera offre des produits de gestion de ces environnements, tel que :

- Cloudera Navigator - protection des données et gouvernance,
- Cloudera Manager – gestion des environnements Hadoop.

L'entreprise Cloudera offre aussi :

- les services professionnels,
- l'option d'exécuter l'environnement dans un modèle infonuagique (Amazon Web Services AWS),
- un service de formation pour les développeurs, les administrateurs et les analystes de données. Cette formation peut être offerte en mode de classe physique, virtuelle, en ligne sur l'internet, etc.

### 7.3.3 Formation offerte

Lors de la lecture de la section de la revue de la littérature sur BigData, il est facile de conclure que l'objectif primaire ou la force primaire d'une solution BigData est l'analyse de données. Il est aussi évident que le BigData est une solution qui peut être utilisée pour résoudre beaucoup d'autres problèmes. Comme toutes autres solutions informatiques, l'exploitation des solutions BigData requiert différentes expertises, tel que :

- Administration de système,
- Programmation,
- Expertise du domaine, tel qu'analyste de données.

Comme mentionné dans le paragraphe précédent, l'entreprise Cloudera offre un service de formation qui reflète les expertises mentionnées précédemment. Une consultation du site web de Cloudera nous offre les formations suivantes. La formation est divisée en trois champs d'expertise:

- Développement logiciel,
- Administration de système,

- Analyse de données.

## Follow the Developer Path

- Developer Training for MapReduce
  - Learn to code and write MapReduce programs for production
  - Master advanced API topics required for real-world data analysis
- Developer Training for Spark and Hadoop I
  - Learn Apache Spark and how it integrates with the entire Hadoop ecosystem
  - How data is distributed, stored, and processed in a Hadoop cluster
- Designing & Building Big Data Applications
  - Build converged applications using multiple processing engines
  - Develop solutions using components across the enterprise data hub
- Spark Training
  - Combine batch and stream processing with interactive analytics
  - Optimize applications for speed, ease of use, and sophistication
- Search Training
  - Bring scalable, flexible indexing to Hadoop with Apache Solr
  - Integrate powerful, real-time queries with external applications
- HBase Training
  - Design schemas to minimize latency on massive data sets
  - Scale hundreds of thousands of operations per second
- Introduction to Data Science »
  - Implement recommenders and data experiments
  - Draw actionable insights from analysis of disparate data

(Hadoop Developer Path, s.d.)

## Follow the Administrator Path

- Administrator Training
  - Configure, install, and monitor clusters for optimal performance
  - Implement security measures and multi-user functionality
- Cloudera Manager Training
  - Use Cloudera Manager to speed deployment and scale the cluster
  - Learn which tools and techniques improve cluster performance
- HBase Training »
  - Implement massively distributed, columnar storage at scale
  - Enable random, real-time read/write access to all data

(Hadoop Administrator Path, s.d.)

**Follow the Data Analyst Learning Path**

- Hadoop Essentials
  - Create value with Hadoop alongside existing technologies
  - Learn how to analyze massive amounts of multi-structured data
- Data Analyst Training
  - Apply SQL and scripting languages to much larger data sets
  - Master advanced techniques that boost Hadoop accessibility
- Introduction to Data Science
  - Build industry-specific data platforms to drive opportunity
  - Use data to reduce costs, increase profit, and retain customers

(Hadoop Data Analyst Path, s.d.)

L'objectif de la présentation des programmes de formation de Cloudera est de démontrer que BigData est un sujet vaste. Il y a de toute évidence trop de matière pour l'objectif de ce projet. Je me limite donc au niveau d'introduction sur les aspects d'administration de système et de la programmation.

### 7.3.4 Survol de la machine virtuelle « Express »

La machine virtuelle Cloudera Express est basée sur le système d'exploitation Linux, plus précisément CentOS version 6.4 64-bit et est disponible pour les hyperviseurs VMWare, VirtualBox et KVM. La machine virtuelle est téléchargeable en format ZIP (4Go) et a une grosseur de 7Go sur disque à la première utilisation.

J'avais à ma disponibilité deux serveurs avec un hyperviseur:

- VMWare ESX/i
- Microsoft Hyper-V

J'ai installé une copie de la VM Cloudera Express sur chacun des serveurs. Il est évident que j'ai été obligé de faire une conversion de la machine virtuelle pour la faire fonctionner sur Hyper-V. J'ai donc utilisé le logiciel « Microsoft Virtual Machine Converter » pour migrer cette machine virtuelle vers Hyper-V.

La machine virtuelle sur Hyper-V :

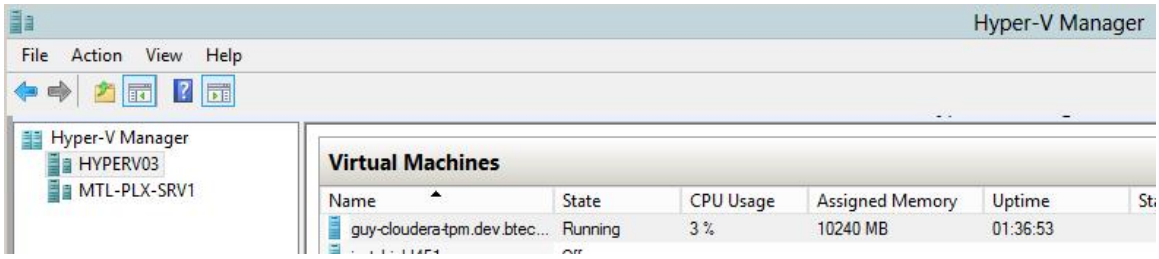


Figure 9 - capture-écran Hyper-V

La machine virtuelle sur VMWare ESX/i :

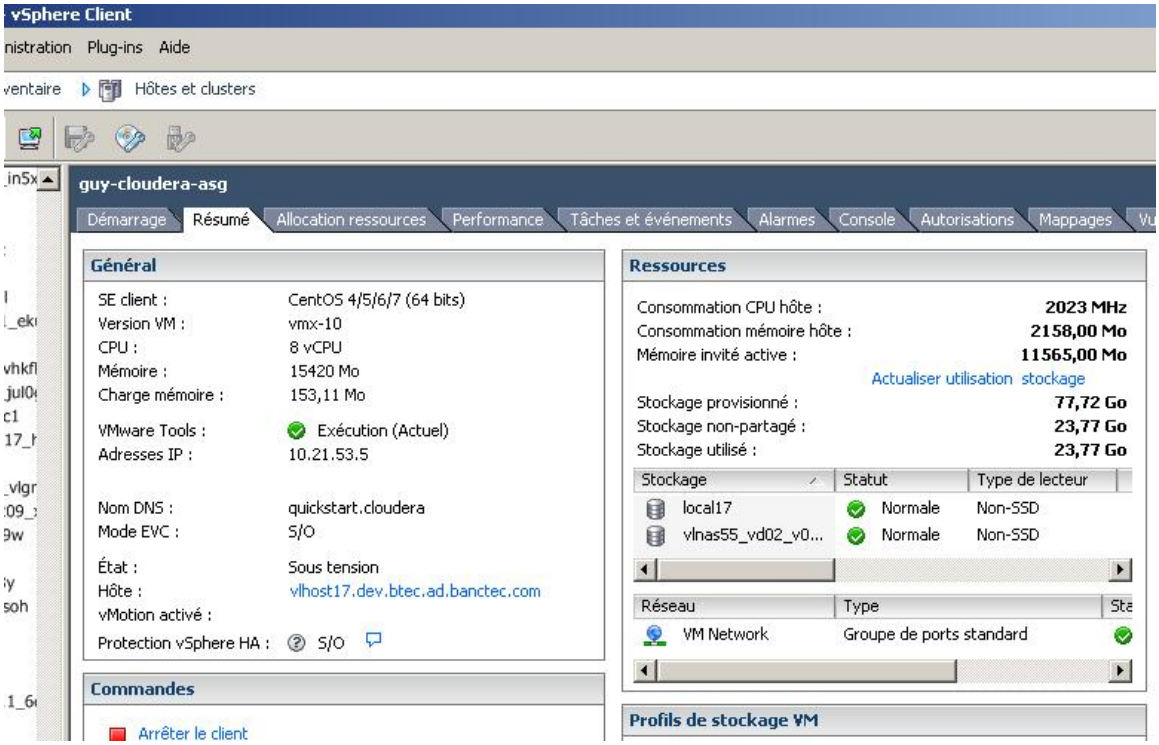


Figure 10 - capture d'écran - VMWare

La console/premier écran présenté à l’usager :

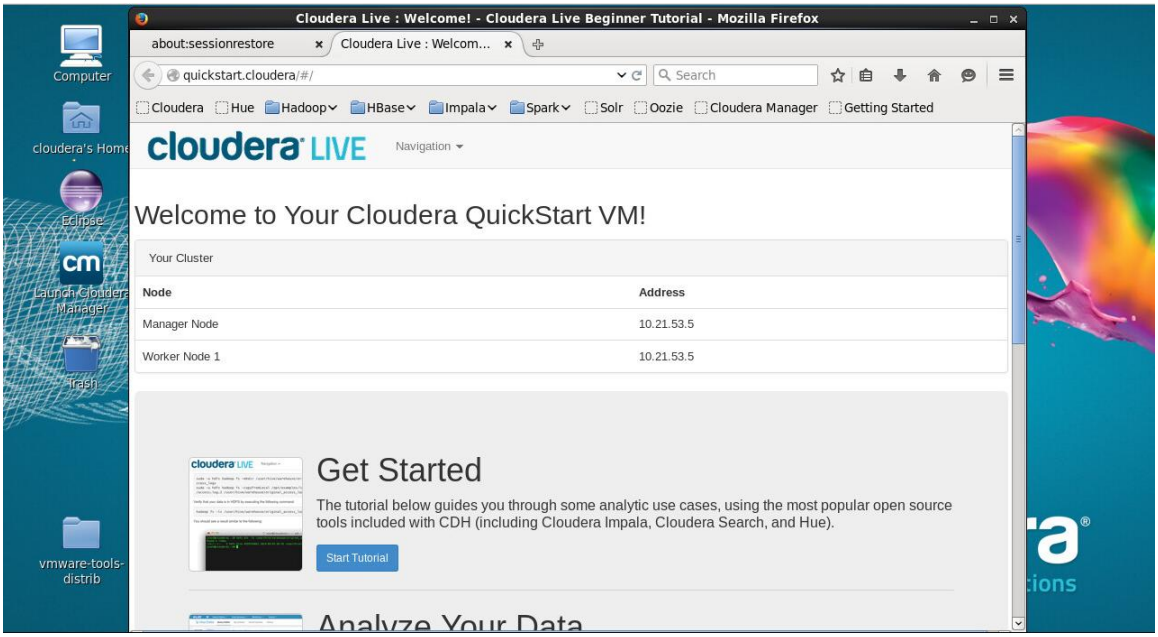


Figure 11 - capture d'écran Cloudera

### 7.3.5 Logiciels BigData installés sur la machine virtuelle

Pour voir «un peu sous le capot» et commencer à comprendre le contenu de la machine virtuelle, je voulais voir les logiciels installés qui ont été nécessaires pour créer un environnement BigData.

Pour faire ceci, j'ai exécuté la commande suivante :

yum list installed

YUM est l'installateur de logiciel sur Fedora/Red Hat/CentOS. Ceci m'a donné une liste de tous les logiciels installés dans l'environnement Centos. J'ai éliminé les logiciels standards de CentOS, et isolé seulement les logiciels que je ne reconnais pas. Il y en a plusieurs.

**Installed Packages**

```
ORBit2.x86_64
avro-libs.noarch
avro-tools.noarch1.7.6+cdh5.4.0+87-1.cdh5.4.0.p0.46.el6
bc.x86_641.06.95-1.el6
bigtop-jsvc.x86_64
bigtop-tomcat.noarch
bigtop-utils.noarch
cairo.x86_64
Mirror/6.4 cloog-ppl.x86_64
cloudera-manager-agent.x86_64
cloudera-manager-daemons.x86_64
cloudera-manager-server.x86_64
crunch.noarch
flume-ng.noarch
flume-ng-agent.noarch
hadoop.x86_64
hadoop-0.20-mapreduce.x86_64
hadoop-client.x86_64
hadoop-conf-pseudo.x86_642.6.0+cdh5.4.0+521-1.cdh5.4.0.p0.59.el6
hadoop-doc.x86_642.6.0+cdh5.4.0+521-1.cdh5.4.0.p0.59.el6
hadoop-hdfs.x86_64
hadoop-hdfs-datanode.x86_64
hadoop-hdfs-fuse.x86_64
hadoop-hdfs-journalnode.x86_64
hadoop-hdfs-namenode.x86_64
hadoop-hdfs-secondarynamenode.x86_64
hadoop-httpfs.x86_64
hadoop-kms.x86_642.6.0+cdh5.4.0+521-1.cdh5.4.0.p0.59.el6
hadoop-libhdfs.x86_64
hadoop-libhdfs-devel.x86_64
hadoop-mapreduce.x86_64
hadoop-mapreduce-historyserver.x86_64
hadoop-yarn.x86_64
hadoop-yarn-nodemanager.x86_64
hadoop-yarn-proxyserver.x86_64
hadoop-yarn-resourcemanager.x86_64
hbase.x86_64
hbase-master.x86_64
hbase-regionserver.x86_641.0.0+cdh5.4.0+131-1.cdh5.4.0.p0.61.el6
hbase-rest.x86_641.0.0+cdh5.4.0+131-1.cdh5.4.0.p0.61.el6
hbase-solr.noarch1.5+cdh5.4.0+49-1.cdh5.4.0.p0.48.el6
hbase-solr-doc.noarch
hbase-solr-indexer.noarch1.5+cdh5.4.0+49-1.cdh5.4.0.p0.48.el6
hbase-thrift.x86_64
hive.noarch
hive-hbase.noarch1.1.0+cdh5.4.0+103-1.cdh5.4.0.p0.56.el6
hive-hcatalog.noarch
hive-jdbc.noarch
hive-metastore.noarch
hive-server2.noarch
hive-webhcat.noarch
hue.x86_64
hue-beeswax.x86_64
hue-common.x86_643.7.0+cdh5.4.0+1145-1.cdh5.4.0.p0.58.el6
hue-hbase.x86_64
hue-impala.x86_643.7.0+cdh5.4.0+1145-1.cdh5.4.0.p0.58.el6
hue-pig.x86_64
hue-plugins.x86_64
hue-rdbms.x86_64
hue-search.x86_643.7.0+cdh5.4.0+1145-1.cdh5.4.0.p0.58.el6
hue-security.x86_64
hue-server.x86_643.7.0+cdh5.4.0+1145-1.cdh5.4.0.p0.58.el6
hue-spark.x86_64
hue-sqoop.x86_64
hue-zookeeper.x86_64
impala.x86_64
impala-catalog.x86_64
impala-server.x86_64
impala-shell.x86_64
impala-state-store.x86_642.2.0+cdh5.4.0+0-1.cdh5.4.0.p0.75.el6
impala-udf-devel.x86_64
jasper-libs.x86_64
kite.noarch
mysql.x86_64
```

On retrouve les logiciels inscrits dans la machine virtuelle BigData de Cloudera. Cependant, nous retrouvons aussi des librairies CORBA, plusieurs librairies de compression, le système de base de données MySql, et plusieurs autres. Il est évident que cette liste représente la force de Cloudera : l'intégration des différents logiciels et composantes nécessaires pour rouler un environnement BigData complet, incluant les outils de gestion. Cependant, pour le néophyte qui veut apprendre le sujet, ce genre d'installation complète est difficile, sinon impossible à faire sans une courbe d'apprentissage très longue et ardue.

### 7.3.6 Tutoriel de Cloudera

La machine virtuelle offerte par Cloudera offre un tutoriel divisé en quatre parties. Le tutoriel est présenté dans une série de pages web située à l'intérieur de la machine virtuelle, donc il n'est pas nécessaire que celle-ci soit branchée à l'internet.

1. Tutoriel section #1 - Intégration avec des systèmes existants de base de données et l'ingestion des données existante
2. Tutoriel section #2 - Interrogation des données structurées
3. Tutoriel section #3 - Relié les données structurées avec les données non structurées
4. Tutoriel section #4 - Analyses avancées avec l'utilisation de " Spark "

#### 7.3.6.1 Tutoriel section #1 - Intégration avec des systèmes existants de base de données et l'ingestion des données existante

La première section du tutoriel démontre l'utilisation de l'outil Apache Scoop (Apache Foundation, s.d.) pour migrer les données d'une base de données traditionnelle vers Hadoop. Pour cet exercice, une base de données MySQL est utilisée. Le schéma de la base de données est relativement simple :

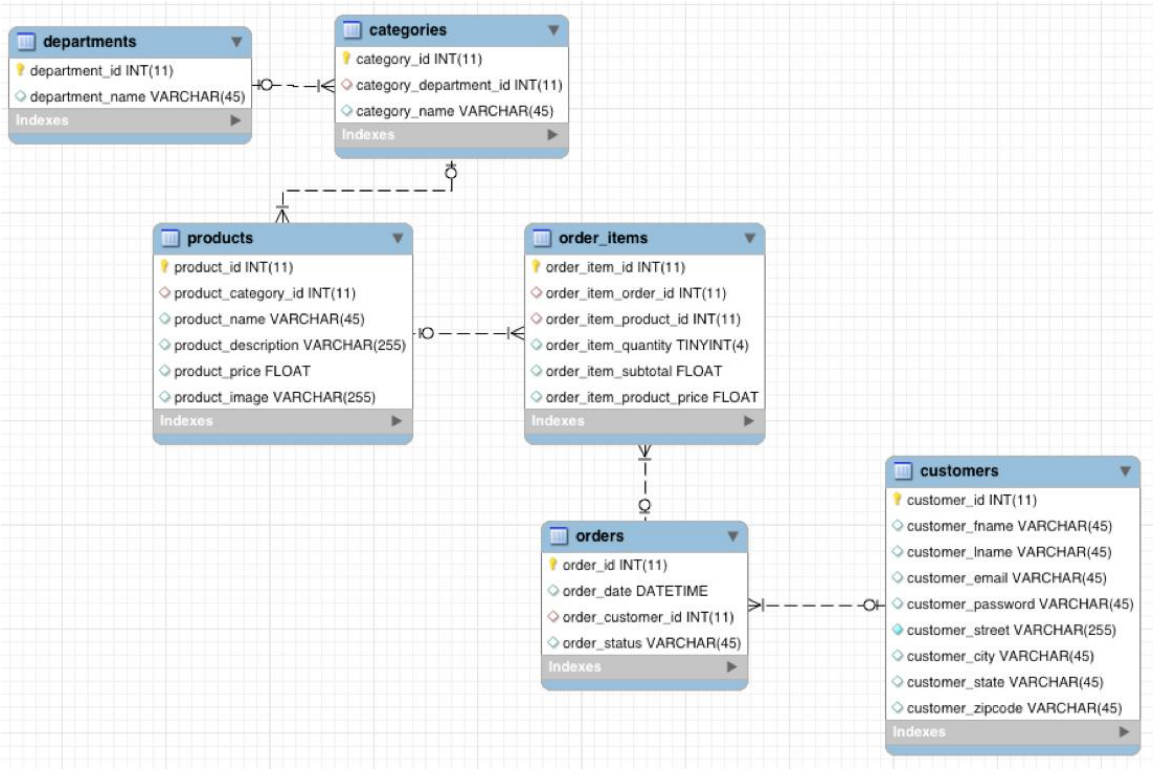


Figure 12 - schéma base de données tutoriel Cloudera (Cloudera)

**Un sommaire du logiciel Sqoop** :

L'outil Sqoop utilise la fonctionnalité « MapReduce » du logiciel Hadoop pour exécuter le transfert de données entre les systèmes de base de données relationnelle et Apache Hadoop. Il est d'une efficacité très avancé avec des transferts en parallèle, réplication et fiabilité. (Apache Foundation, s.d.) (Cloudera)

Le tutoriel inclut les étapes pour exécuter Sqoop, vérifier le résultat et préparer les informations nécessaires pour la section 2 du tutoriel. Un premier bémol sur le tutoriel : il y a quelques technologies qui sont simplement mentionnées sans fournir de détail. Je note immédiatement la mention de « Apache Avro – Hadoop optimized file format ». Cependant, lors de la recherche sur Apache Avro, sa documentation la décrit comme étant de système de sérialisation avec schéma. De toute évidence, il y a une étude plus approfondie qui sera nécessaire après ce projet lors de mon cheminent d'apprentissage de cette technologie.

#### 7.3.6.2 Tutoriel section #2 - Interrogation des données structurées

L'objectif de la section 2 est l'interrogation de nos données dans la base de données Hadoop. On nous présent l'outil HUE, qui nous permet d'utiliser une interface web pour faire nos interrogations. H, présente avec plusieurs choix d'éditeurs d'interrogations, tel que Hive, Impala, Pig, etc., et des outils de navigation de données, comme HBase (Hadoop), ZooKeeper (configurations), etc.


Figure 13 - capture d'écran - Cloudera query editors

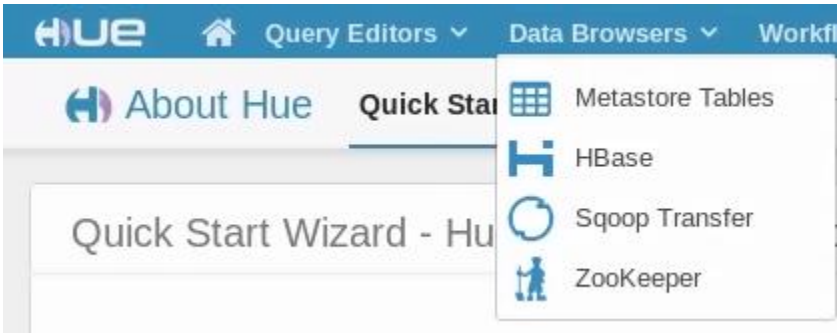


Figure 14 - capture d'écran - Cloudera data browsers

Chaque outil est utilisé pour une tache spécialisé. L'exemple donné dans le tutoriel est que l'outil Hive est l'outil préféré pour des interrogations avec le langage de définition de données LDD (Data definition langage DDL en anglais), mais que l'outil Impala est préférable pour des interrogations interactives.

L'outil Impala nous permet d'utiliser la syntaxe d'interrogation SQL bien connue, tel que l'exemple ci-dessous nous démontre :

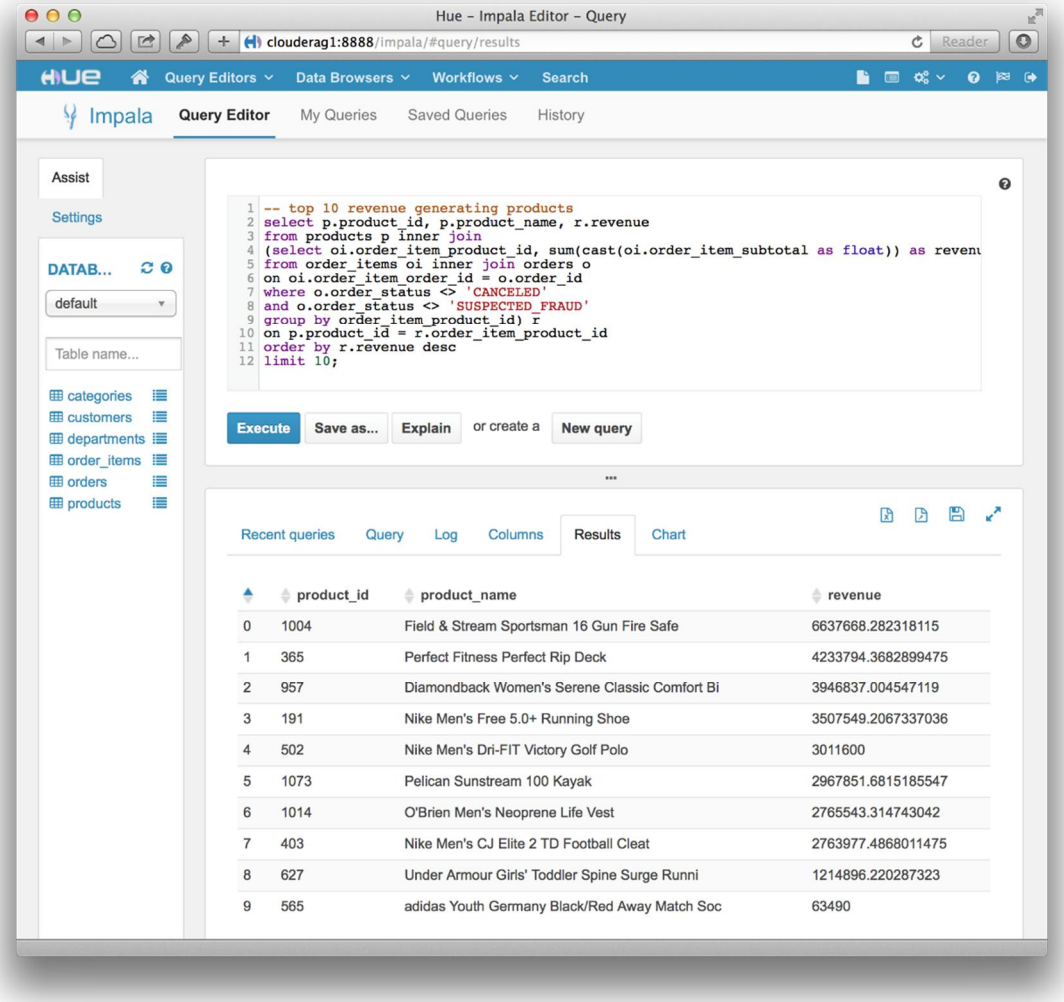


Figure 15 - capture d'écran - Cloudera Impala

#### 7.3.6.3 Tutoriel section #3 - Relié les données structures avec les données non structurées

La section 3 de ce tutoriel nous présente l'outil « Apache Flume » pour faire l'ingestion de données non structurées dans Hadoop, telles que les journaux d'accès d'un site web. Pour faire suite à cette ingestion, le tutoriel se retourne vers l'outil Impala pour refaire une recherche avec

les deux genres de données. L'exemple présenté est que les journaux du serveur web démontrent que les clients accèdent à une page qui a une erreur et donc ne peuvent pas l'acheter. La force de Flume et Impala est que cette recherche est facile, rapide et intégrée. Le tutoriel suggère aussi que sans l'utilisation de ces outils, la possibilité que cette erreur ne soit pas découverte pour une longue période est possible, ce qui entraine une perte de revenu pour l'entreprise.

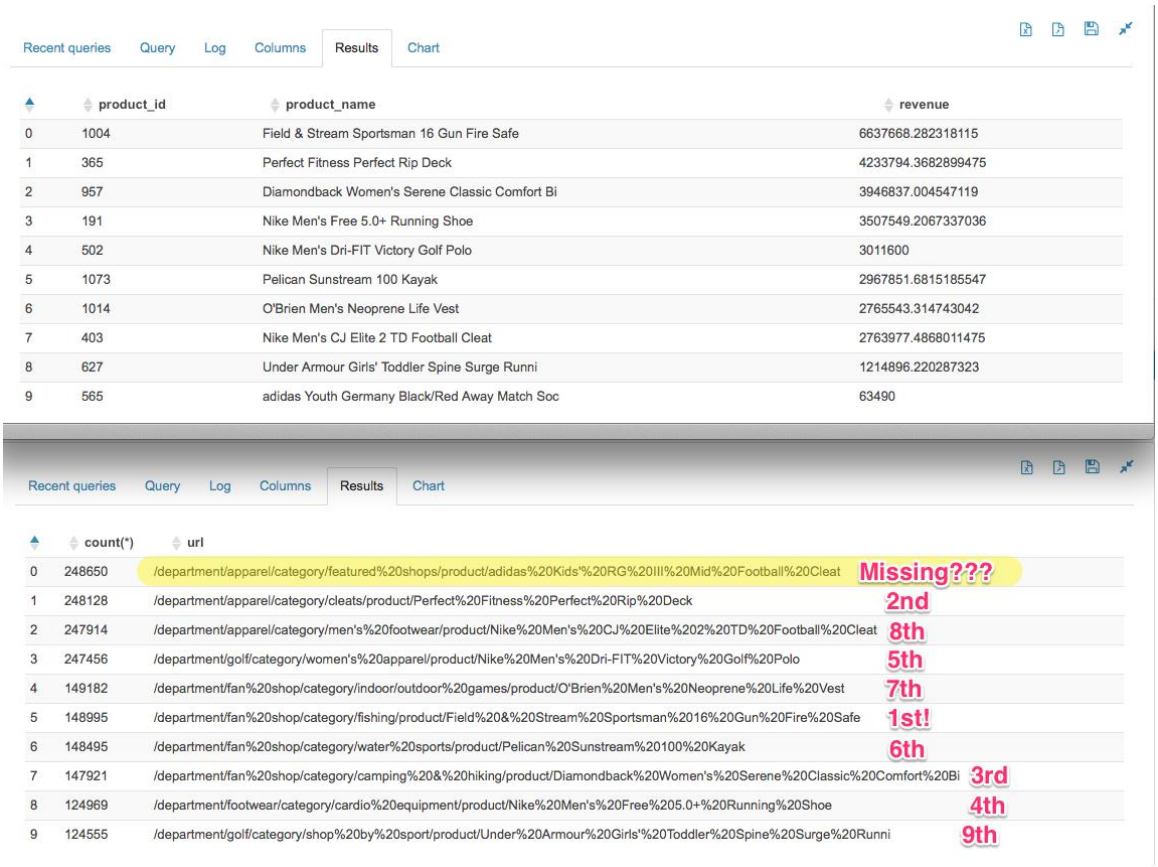


Figure 16 - capture d'écran - Cloudera Flume

#### 7.3.6.4 Tutoriel section #4 - Analyses avancées avec l'utilisation de « Spark »

La section 4 du tutoriel nous présente l'outil "Apache Spark » et sa capacité de faire des analyses rapide sur la relation entre des objets.

L'objectif du tutoriel est de démontrer que la création d'une interrogation analytique est facile et rapide, et que celle-ci a aussi une rapidité d'exécution.

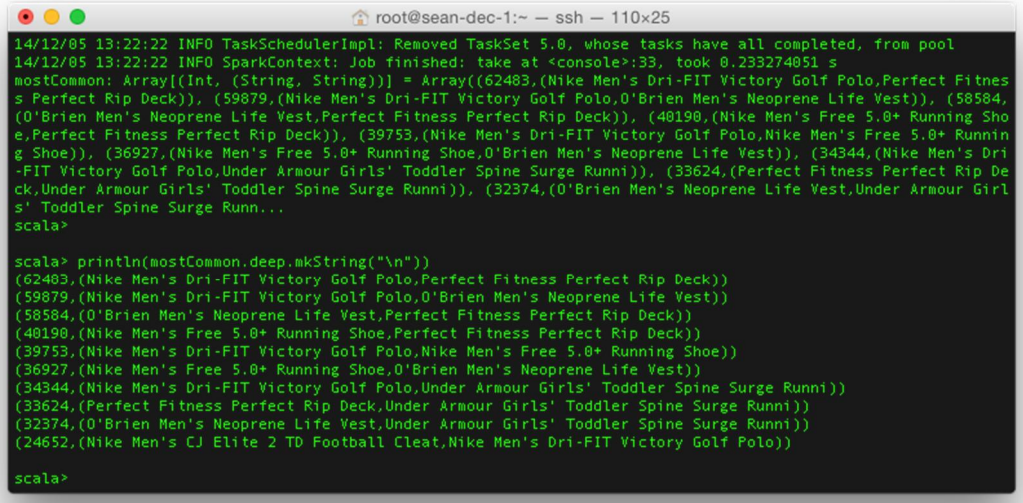


Figure 17 - capture d'écran - Cloudera Spark

#### 7.3.6.5 Appréciation du tutoriel de Cloudera

Ce tutoriel présente beaucoup de matière de façon rapide, visuelle et concise. Elle accomplit très bien son objectif pour piquer notre curiosité et intérêt pour ce sujet, mais elle démontre aussi que cette solution et ses technologies sont d'une ampleur significative.

### 7.4 Tutoriel #2 – Installation de HBase

L'environnement Cloudera et ses tutoriels démontrent un environnement « big data/hadoop » avec tous les outils de gestion, interrogations et autres. Le tutoriel est plutôt orienté pour nous donner une vue d'ensemble et d'environnement multiserveur. Pour ma deuxième installation, je vais me concentrer strictement sur Apache Hadoop (Hbase et MapReduce), pour descendre à un niveau un peu plus technique.

### 7.4.1 Apache HBase

HBase (Apache Foundation, s.d.) est un projet de la fondation Apache. Le site de la fondation Apache n'est pas disponible en français, mais la description de HBase en français utilisé dans Wikipedia est :

> HBase est un système de gestion de bases de données non relationnelles distribuées, écrit avec le langage Java, disposant d'un stockage structuré pour les grandes tables.
> HBase est inspirée des publications de Google sur BigTable. Comme BigTable, elle est une base de données orientée colonnes. Basées sur une architecture maître/esclave, les bases de données de ce type sont capables de gérer d'énormes quantités d'informations (plusieurs milliards de lignes par table). (Wikipedia, s.d.)

### 7.4.2 Attention aux néophytes!

Dans le guide officiel de HBase (Apache Foundation, s.d.), l'un des premiers paragraphes offre les conseils suivants pour ceux qui s'aventurent sur ce sujet pour la première fois :

- Les systèmes distribués sont difficiles,
- Pour faire bien fonctionner ce genre de système, vous avez besoin des connaissances disparates et hétérogènes sur une grande quantité de sujets dans les domaines de l'équipement informatique, logicielles et réseautiques,
- Votre installation peut avoir des problèmes causés par une foule de problèmes mystérieux, possiblement causés par des bogues dans HBase, des erreurs de configurations, des problèmes/bogues dans le pilote de votre carte réseau, pour en nommer quelques-uns,
- bienvenue, c'est un monde intéressant.

### 7.4.3 Tutoriel choisi – HBase, installation locale et unique sur un ordinateur portatif avec Windows 7

Dans mes démarches de mon premier tutoriel, il a été noté que la machine virtuelle de Cloudera a demandé une quantité de ressources assez cossue. J'ai été capable de me procurer les ressources nécessaires chez mon employeur. Cependant, ce genre d'installation n'est pas nécessairement à la portée de la main du néophyte typique et possiblement décourager celui-ci de se lancer dans cette technologie. Pour mon deuxième tutoriel, j'ai choisi d'approfondir mes connaissances spécifiquement au sujet du logiciel et base de données HBase, avec les contraintes d'utiliser un ordinateur portatif typique avec 4 CPUs et 4Gb de mémoire, installé avec MS-Windows 7. L'objectif étant que je sois capable de travailler sur HBase lors de mes déplacements quotidiens par train de banlieue pour le travail.

### 7.4.4 Les prérequis

Même si HBase est conçue pour les environnements *nix, le code source est écrit en Java, qui peut rouler presque n' importe où, incluant Windows. Cependant, tous les scripts sont écrits en langage de script Unix (SH, BASH), ce qui n'est pas disponible sur Windows de façon native.

La documentation (Apache Foundation, s.d.) nous met en garde au sujet des problèmes des différences entre la façon de définir les chemins d'accès entre Windows et Unix. Ceci inclut :

- L'utilisation de / ou \
- L'utilisation de la nomenclature Windows (c:\utilisateurs\gbertrand), celle de *nix (/usr/local/…) et même celle de CYGWIN (/cygdrive/d/hbase/…)
- La différence et l'utilisation des chemins de système d'exploitation et celle de Java (file:///....)

Donc la documentation (Apache Foundation, s.d.) demande l'installation de trois logiciels supplémentaires :

- Java (JRE, mais préférable JDK),

- Cygwin (environnement Linux pour Windows),
- SSH (OpenSSH, tcp_wrappers, diffutils, zlib).

En réalité, les utilitaires SSH sont des composantes qui sont installées par ajout dans l'environnement Cygwin.

Java est installé

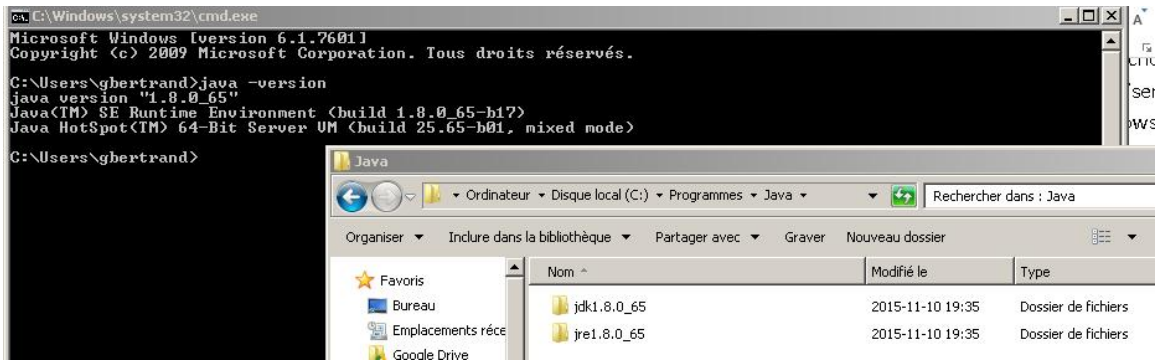


Figure 18 - capture d'écran - installation Java

Cygwin est disponible :

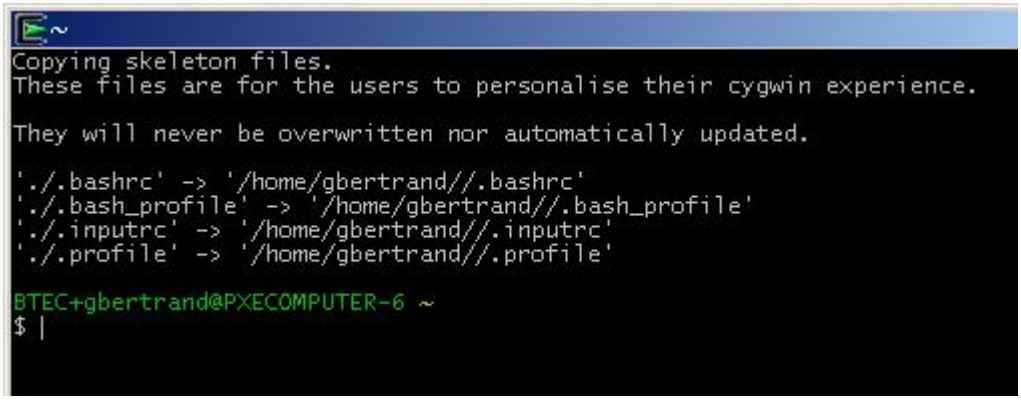


Figure 19 - capture d'écran - Cygwin

SSH, diff, gzip(zlib) (à l'intérieur de Cygwin)

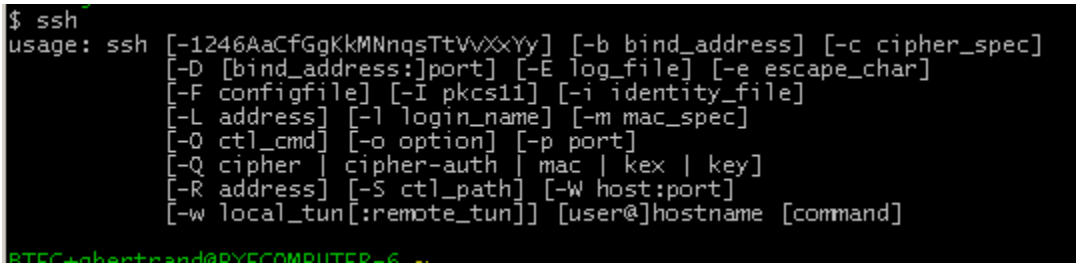


Figure 20 - capture d'écran - SSH

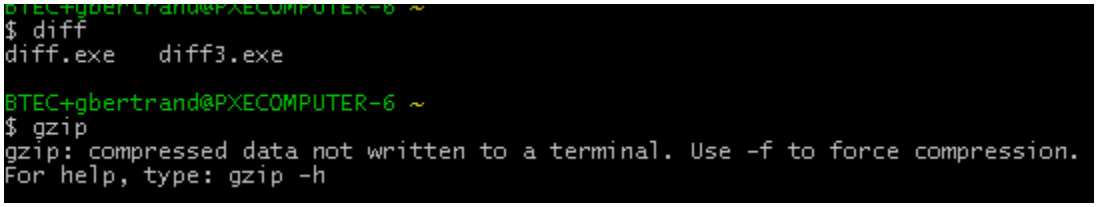


Figure 21 - capture d'écran – diff

### 7.4.5 Installation de HBase

HBase est fournie dans un fichier compressé format GZIP. Il suffit de la décompresser. Il est à noter que j'ai ajouté une étape supplémentaire ci-dessous : j'ai exécuté la commande GUNZIP avant TAR pour bien démontrer le processus.

Cependant, la commande TAR est capable d'utiliser les librairies de décompression directement, donc la commande GUNZIP n'était pas nécessaire. Mais mon expérience m'a appris que cette petite fonctionnalité de la commande TAR n'est pas nécessairement bien connue.

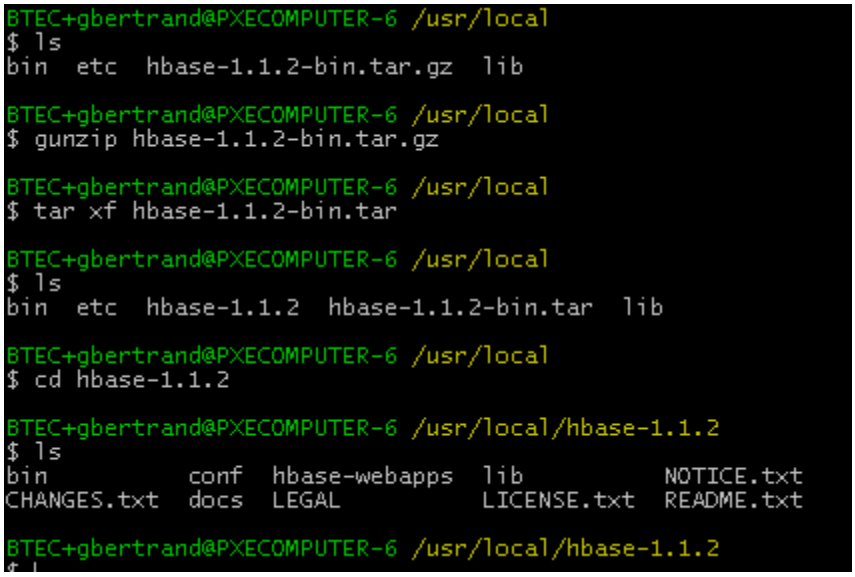


Figure 22 - capture d'écran - HBase

### 7.4.6 Installation et configuration de SSH

Pour faire suite à la série de commande pour configurer le serveur SSHD, la vérification ce fait comme suit :

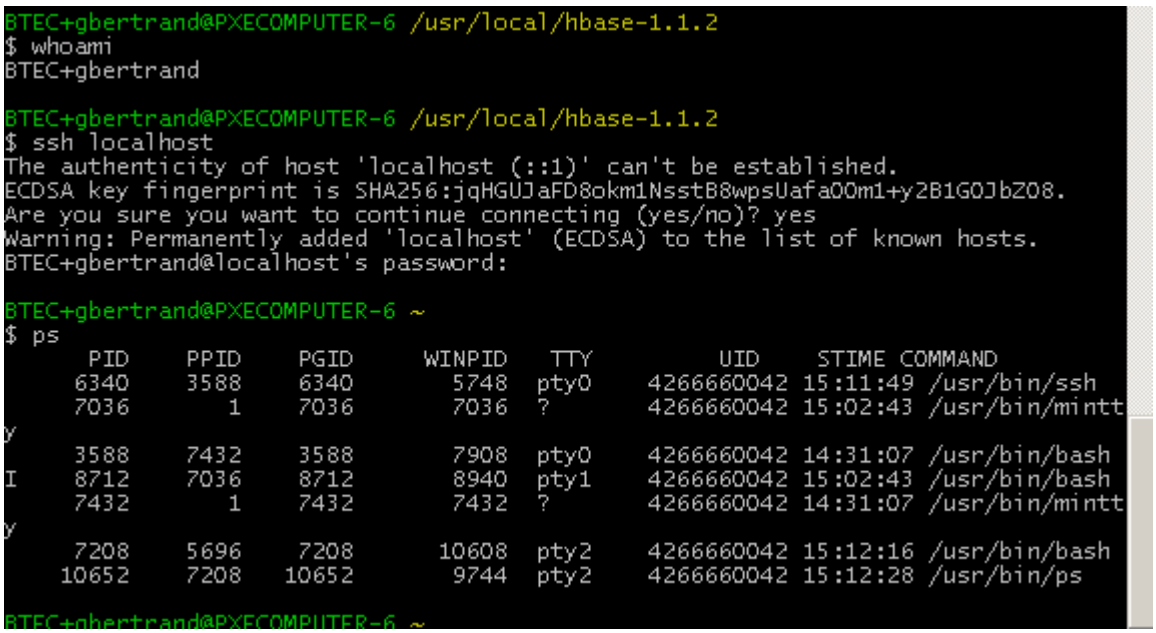


Figure 23 - capture d'écran - SSH configuration

### 7.4.7 Configuration de HBase

La documentation demande la modification du fichier de configuration de HBase « .conf/hbase-env.sh » et « ./conf/hbase-default.xml ».
Le premier changement que j'ai fait est la configuration du répertoire pour les journaux de Hbase, que j'ai redirigé vers le disque D :

```
# Where log files are stored.  $HBASE_HOME/logs by default.
# export HBASE_LOG_DIR=${HBASE_HOME}/logs
export HBASE_LOG_DIR=/cygdrive/d/hbase/logs
```

Figure 24 - capture d'écran - config HBase 1

Cependant, l'installation de cette version de HBase n'inclut pas le deuxième fichier « hbase-default.xml ».

```
BTEC+gbertrand@PXECOMPUTER-6 /usr/local/hbase-1.1.2/conf
$ ls
hadoop-metrics2-hbase.properties  hbase-env.cmd  hbase-env.sh  hbase-policy.xml  hbase-site.xml  log4j.properties  regionservers
```

Figure 25 - capture d'écran - config HBase 2

Ce fichier est le fichier de configuration pour HBase et est important, car je désirai que les données de HBase soit sur le disque D :, car mon disque C : n'avait pas assez d'espace. Une petite recherche Google m'a permis de trouver le code source de HBase et ce fichier (https://github.com/apache/hbase/blob/master/hbase-common/src/main/resources/hbase-default.xml). J'ai modifié les valeurs pour les variables suivantes :

| Variables | Nouvelle valeur |
|---|---|
| Hbase.tmp.dir | D:/hbase/tmp |
| Hbase.rootdir | file:///d:/hbase/hbase |

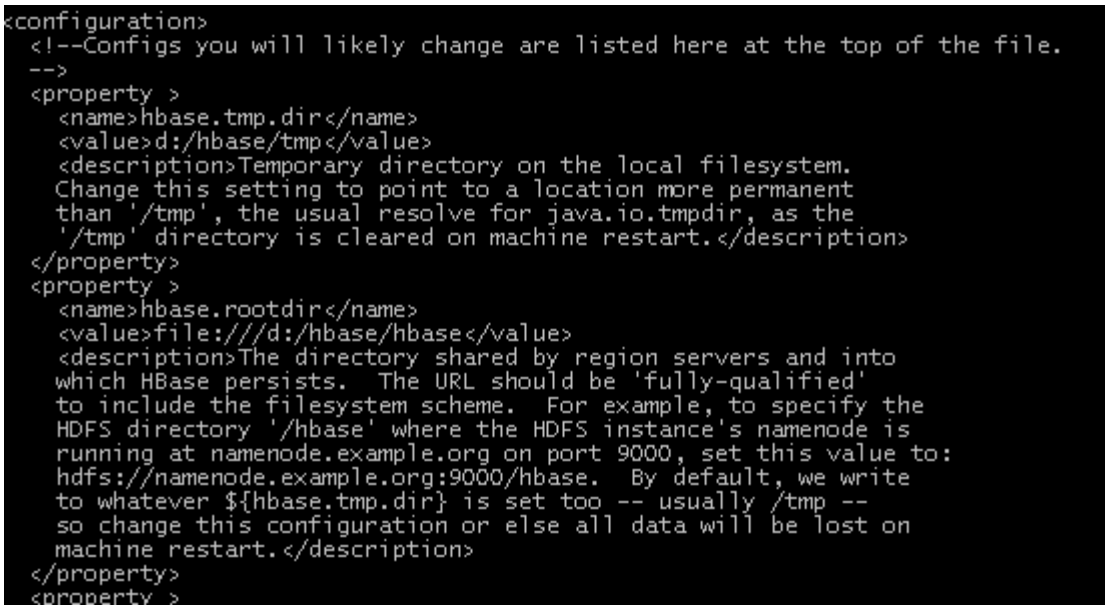


Figure 26 - capture d'écran - HBase configuration 3

J'ai aussi modifié la variable hbase.zookeeper.quorum telle que suggérée. Il semble que Cygwin a possiblement des problèmes d'interprétation de « localhost ».

| Variable | Ancienne valeur | Nouvelle valeur |
|---|---|---|
| Hbase.zookeeper.quorum | Localhost | 127.0.0.1 |

### 7.4.8 Winutils

HBase sur Windows utilise quelques utilitaires compilés en mode natif pour MS-Windows. Ces utilitaires ne sont pas inclus dans la distribution standard de HBase. Il faut faire un téléchargement de ces utilitaires du site web de HBase et les copier dans le sous-répertoire BIN de l'installation HBase.

### 7.4.9 Échec…

Ces configurations n'ont jamais fonctionné. J'ai reçu une quantité énorme de messages d'erreur du genre « not found », et « null path ». J'ai même été obligé de modifier le fichier

« hbase-default.xml » pour y inscrire la version courante de Hbase. Il devient très évident que la page de référence sur le site Apache n'est possiblement plus à date, et qu'aussi que le fichier hbase-default.xml n'est plus utilisé dans les dernières versions de Hbase.

### 7.4.10 Utilisation des fichiers CMD au lieu de SH

La dernière version de HBase inclus aussi des fichiers *.CMD à côté des fichiers *.SH. Dans la section 3.4, j'ai mentionné que HBase était écrit avec le langage Java, ce qui devrait rouler sur n'importe quel système d'exploitation. Le problème est spécifiquement les scripts de démarrage et arrêt, qui ne fonctionnent que sur *nix. Cependant, on peut retrouver des fichiers *.CMD, qui sont des fichiers de scripts pour l'environnement Windows.

### 7.4.11 Première aperçue des fichiers CMD

Une revue des fichiers *.CMD démontre qu'il n'y presque pas de modification à faire, sauf pour l'endroit de mon installation Java, que je modifie pour :

```
set JAVA_HOME="C:\program files\Java\jre1.8.0_65"
```

Cependant, j'ai immédiatement des erreurs, même avec l'utilisation des apostrophes pour entourer le chemin avec l'espace entre « program » et « files ». Je dois utiliser la convention de MS-DOS avec les noms de fichiers en format 8.3. La commande pour trouver le nom est 'dir /x' :

```
Démarrer CMD
C:\> c :     (si vous n'êtes pas sur le disque c : )
C:\> cd \ (pour s'assurer que nous sommes sur le répertoire principal
C:\> dir /x
…
2015-11-10  20:52    <REP>          PROGRA~1     Program Files
```

Donc je modifie cette ligne (vers la ligne 22) dans le fichier hbase-env.cmd :

```
set JAVA_HOME="C:\progra~1\Java\jre1.8.0_65"
```

### 7.4.12 Une dernière modification

Le démarrage de HBase (avec la commande bin\start-hbase.cmd) ne fonctionne pas encore. Je reçois encore des messages d'erreurs. Une vérification des journaux indique que la variable HADOOP_HOME n'est pas configurée/vide.

Une recherche dans tous les fichiers CMD ne démontre nullement la déclaration de cette variable.

Donc je dois la déclarer de la façon suivante :

Set HADOOP_HOME = d:\hbase\hbase-1.1.2

En ensuite la commande bin\start-hbase.cmd.

**Succès**!!!

### 7.4.13 HBase en action!

```
2015-11-15 12:26:53,755 INFO  [main] util.VersionInfo: HBase 1.1.2
2015-11-15 12:26:53,771 INFO  [main] util.VersionInfo: Source code repository git://hw11397.local/Volumes/hbase-1.1.2RC2/hbase revision=cc2b70cf03e3378
800661ec5cab11eb43fafe0fc
2015-11-15 12:26:53,771 INFO  [main] util.VersionInfo: Compiled by ndimiduk on Wed Aug 26 20:11:27 PDT 2015
2015-11-15 12:26:53,771 INFO  [main] util.VersionInfo: From source with checksum 73da41f3d1b867b7aba6166c77fafc17
2015-11-15 12:26:53,958 INFO  [main] master.HMasterCommandLine: Starting a zookeeper cluster
2015-11-15 12:26:53,989 INFO  [main] serveur.ZooKeeperServer: Server environment:zookeeper.version=3.4.6-1569965, built on 02/20/2014 09:09 GMT
2015-11-15 12:26:53,989 INFO  [main] server.ZooKeeperServer: Server environment:host.name=PXECOMPUTER-6.ca.ad.banctec.com
2015-11-15 12:26:53,989 INFO  [main] server.ZooKeeperServer: Server environment:java.version=1.8.0_65
2015-11-15 12:26:53,989 INFO  [main] server.ZooKeeperServer: Server environment:java.vendor=Oracle Corporation
2015-11-15 12:26:53,989 INFO  [main] server.ZooKeeperServer: Server environment:java.home=C:\progra~1\Java\jre1.8.0_65
2015-11-15 12:26:53,989 INFO  [main] server.ZooKeeperServer: Server environment:java.class.path=D:\hbase\hbase-1.1.2\bin\..\conf;C:\progra~1\Java\jre1.
8.0_65\lib\tools.jar;D:\hbase\hbase-1.1.2\bin\..;D:\hbase\hbase-1.1.2\bin\..\lib\activation-1.1.jar;D:\hbase\hbase-1.1.2\bin\..\lib\aopalliance-1.0.jar
;D:\hbase\hbase-1.1.2\bin\..\lib\apacheds-i18n-2.0.0-M15.jar;D:\hbase\hbase-1.1.2\bin\..\lib\apacheds-kerberos-codec-2.0.0-M15.jar;D:\hbase\hbase-1.1.2
\bin\..\lib\api-asn1-api-1.0.0-M20.jar;D:\hbase\hbase-1.1.2\bin\..\lib\api-util-1.0.0-M20.jar;D:\hbase\hbase-1.1.2\bin\..\lib\asm-3.1.jar;D:\hbase\hbas
e-1.1.2\bin\..\lib\avro-1.7.4.jar;D:\hbase\hbase-1.1.2\bin\..\lib\commons-beanutils-1.7.0.jar;D:\hbase\hbase-1.1.2\bin\..\lib\commons-beanutils-core-1.
8.0.jar;D:\hbase\hbase-1.1.2\bin\..\lib\commons-cli-1.2.jar;D:\hbase\hbase-1.1.2\bin\..\lib\commons-codec-1.9.jar;D:\hbase\hbase-1.1.2\bin\..\lib\commo
ns-collections-3.2.1.jar;D:\hbase\hbase-1.1.2\bin\..\lib\commons-compress-1.4.1.jar;D:\hbase\hbase-1.1.2\bin\..\lib\commons-configuration-1.6.jar;D:\hb
ase\hbase-1.1.2\bin\..\lib\commons-daemon-1.0.13.jar;D:\hbase\hbase-1.1.2\bin\..\lib\commons-digester-1.8.jar;D:\hbase\hbase-1.1.2\bin\..\lib\commons-e
l-1.0.jar;D:\hbase\hbase-1.1.2\bin\..\lib\commons-httpclient-3.1.jar;D:\hbase\hbase-1.1.2\bin\..\lib\commons-io-2.4.jar;D:\hbase\hbase-1.1.2\bin\..\lib
\commons-lang-2.6.jar;D:\hbase\hbase-1.1.2\bin\..\lib\commons-logging-1.2.jar;D:\hbase\hbase-1.1.2\bin\..\lib\commons-math-2.2.jar;D:\hbase\hbase-1.1.2
\bin\..\lib\commons-math3-3.1.1.jar;D:\hbase\hbase-1.1.2\bin\..\lib\commons-net-3.1.jar;D:\hbase\hbase-1.1.2\bin\..\lib\disruptor-3.3.0.jar;D:\hbase\hb
ase-1.1.2\bin\..\lib\findbugs-annotations-1.3.9-1.jar;D:\hbase\hbase-1.1.2\bin\..\lib\guava-12.0.1.jar;D:\hbase\hbase-1.1.2\bin\..\lib\guice-3.0.jar;D:
\hbase\hbase-1.1.2\bin\..\lib\guice-servlet-3.0.jar;D:\hbase\hbase-1.1.2\bin\..\lib\hadoop-annotations-2.5.1.jar;D:\hbase\hbase-1.1.2\bin\..\lib\hadoop
-auth-2.5.1.jar;D:\hbase\hbase-1.1.2\bin\..\lib\hadoop-client-2.5.1.jar;D:\hbase\hbase-1.1.2\bin\..\lib\hadoop-common-2.5.1.jar;D:\hbase\hbase-1.1.2\bi
n\..\lib\hadoop-hdfs-2.5.1.jar;D:\hbase\hbase-1.1.2\bin\..\lib\hadoop-mapreduce-client-app-2.5.1.jar;D:\hbase\hbase-1.1.2\bin\..\lib\hadoop-mapreduce-c
…
…
…
Beaucoup d'autres textes sans message d'erreur!
```

### 7.4.14 Vérification de l'installation de HBase

Pour vérifier le bon fonctionnement de HBase, j'ai suivi les instructions de la page d'Apache : (The Apache Software Foundation, s.d.)

So it's time to test it.
Start HBase using the command ./bin/start-hbase.sh
When prompted to accept the SSH fingerprint, answer yes.
When prompted, provide your password. Maybe multiple times.
When the command completes, the HBase server should have started.
However, to be absolutely certain, check the logs in the ./logs directory for any exceptions.
Next we start the HBase shell using the command ./bin/hbase shell
We run some simple test commands
Create a simple table using command create 'test', 'data'
Verify the table exists using the command list
Insert data into the table using e.g.
put 'test', 'row1', 'data:1', 'value1'
put 'test', 'row2', 'data:2', 'value2'
put 'test', 'row3', 'data:3', 'value3'
List all rows in the table using the command scan 'test' that should list all the rows previously inserted. Notice how 3 new columns where added without changing the schema!
Finally we get rid of the table by issuing disable 'test' followed by drop 'test' and verified by list which should give an empty listing.

(The Apache Software Foundation, s.d.)

## Mes résultats:

```
D:\hbase\hbase-1.1.2>bin\hbase shell
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.1.2, rcc2b70cf03e3378800661ec5cab11eb43fafe0fc, Wed Aug 26 20:11:27 P
T 2015
hbase(main):001:0> create 'test','data'
0 row(s) in 1.5120 seconds
=> Hbase::Table - test
hbase(main):002:0> list
TABLE
test
1 row(s) in 0.0480 seconds
=> ["test"]
hbase(main):003:0> put 'test', 'row1', 'data:1', 'value1'
0 row(s) in 0.1240 seconds
hbase(main):004:0> put 'test', 'row2', 'data:2', 'value2'
0 row(s) in 0.0000 seconds
hbase(main):005:0> put 'test', 'row3', 'data:3', 'value3'
0 row(s) in 0.0310 seconds
hbase(main):006:0> scan 'test'
ROW                 COLUMN+CELL
 row1               column=data:1, timestamp=1447608544173, value=value1
 row2               column=data:2, timestamp=1447608544205, value=value2
 row3               column=data:3, timestamp=1447608545109, value=value3
3 row(s) in 0.0310 seconds
hbase(main):007:0> disable 'test'
0 row(s) in 2.3250 seconds
hbase(main):008:0> drop 'test'
0 row(s) in 1.2950 seconds
hbase(main):009:0> list
TABLE
0 row(s) in 0.0000 seconds
=> []
hbase(main):010:0> exit
D:\hbase\hbase-1.1.2>
```

Figure 22 - capture d'écran - test HBase

SUCCÈS!!!!

### 7.4.15 Reculer d'un pas

Dans la section précédente, j'ai mentionné que j'ai créé une variable d'environnement HADOOP_HOME. Lors de mes tests avec Cygwin et les scripts *.sh, je n'ai jamais créé cette variable d'environnement. Je suis toujours d'accord avec le principe de la simplification des environnements informatiques, aussi connue par le concept KISS (keep it simple s…..). Je préfère grandement ma réussite sans l'utilisation de Cygwin, SSH et les scripts *.sh.

### 7.4.16 Une autre approche plus facile

Dans mon premier tutoriel, j'ai présenté l'environnement Cloudera. Il est à noter que Cloudera n'est pas la seule entreprise qui offre des environnements Hadoop intégrés. Je note spécifiquement HortonWorks, qui est une entreprise créée par d'anciens employés de Yahoo. Yahoo est un des plus gros utilisateurs de « Hadoop/big data » au monde.
Pourquoi HortonWorks?? Ils sont partenaires avec Microsoft pour l'utilisation de Hadoop sur le système infonuagique Azure. De plus, ils offrent plusieurs options pour l`utilisation de Hadoop sur un poste Windows pour les développeurs, avec installation facile. Je n'ai pas choisi ce chemin plus facile, car comme j'ai précisé dans l'introduction de ce chapitre :

> « avec les contraintes d'utiliser un ordinateur portatif typique avec 4 CPUs et 4Gb de mémoire, installés avec MS-Windows 7. L'objectif étant que je sois capable de travailler sur HBase lors de mes déplacements quotidiens par train de banlieue pour le travail. »

## 7.5 Tutoriel #3 – un programme Java avec HBase

Pour ce troisième tutoriel, je veux écrire/copier du code Java pour faire des opérations dans ma base donnée HBase. Une recherche sur Google me donne beaucoup de choix. Je choisis les sites/pages suivants :

- exemple dans le blogue de Xu Fei, [13]
- MapReduce Tutorial - Apache Hadoop. [14]

### 7.5.1 Exemple #1 dans le blogue de Xu Fei

En réalité, Xu présente 2 exemples dans son blogue. Je l'ai choisie, car les deux exemples démontrent les opérations primaires pour la persistance :

> *The four primary data model operations are Get, Put, Scan, and Delete. Operations are applied via HTable instances.*

(Fei, s.d.)

De plus, les 2 exemples ont beaucoup de commentaires. Une copie des exemples sera ajoutée à l'annexe I et II de ce document.

### 7.5.2 Les librairies (*.jar)

Un des premiers commentaires sur la page est au sujet des librairies qui sont nécessaires pour compiler.

Mais le problème est que cette page a été écrite en 2009 pour la version 0.92.1 de HBase. J'utilise la version 1.1.2. Les librairies ne sont pas les mêmes. Ils semblent que les librairies ont été divisées, car je ne retrouve pas une librairie « hbase-x.x.x.jar », mais plutôt des librairies « hbase-client-x.x.x.jar et hbase-server-x.x.x.jar » (voir ci-dessous):

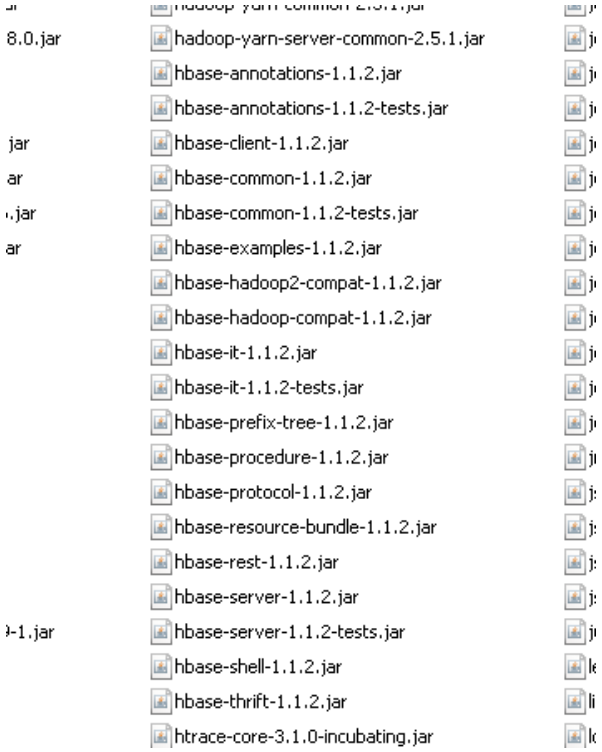


Figure 27 - capture d'écran - liste partielle librairies HBase

L'auteur indique d'inclure toutes les librairies dans ce sous-répertoire. Je ne suis pas d'accord avec la quantité de librairies spécifiées. Mon expérience a démontré que les programmeurs Java ajoutent beaucoup trop de librairies inutilement, simplement pour ne pas avoir à faire l'étape que je vais faire maintenant : le choix du minimum de librairies nécessaires pour compiler. À mon avis, ceci est nécessaire pour la bonne maintenance de code, pour aider la rétro-ingénierie. Trop souvent, on regarde du code Java vieux de quelques années et l'on retrouve un tas de librairies inutiles…possiblement. On ne le sait pas. Pourquoi cette librairie est-elle là? Existe-t-il une dépendance ou simplement une paresse de la dernière personne qui a touché à ce code?

La première étape est la création d'un nouveau projet Eclipse. Je copie le premier exemple : HBaseConnector. Il y a immédiatement des erreurs, car il manque des librairies.

```
1  import java.io.IOException;
2
3  import org.apache.hadoop.conf.Configuration;
4  import org.apache.hadoop.hbase.HBaseConfiguration;
5  import org.apache.hadoop.hbase.client.Get;
6  import org.apache.hadoop.hbase.client.HTable;
7  import org.apache.hadoop.hbase.client.Put;
8  import org.apache.hadoop.hbase.client.Result;
9  import org.apache.hadoop.hbase.client.ResultScanner;
10 import org.apache.hadoop.hbase.client.Scan;
11 import org.apache.hadoop.hbase.util.Bytes;
12
13 public class HBaseConnector {
14     public static void main(String[] args) throws IOException {
15
```

Figure 28 - capture d'écran - projet Eclipse librairies

Succès!  J'ai été capable de réduire le nombre de librairies à 4 (de 9 dans l'exemple)


Figure 29 - capture d'écran - premier ajout de librairie

### 7.5.3 Première exécution de l'exemple #1

Même s'il n'y a plus d'erreur de dépendance avec le code et ses « imports », les librairies utilisées ont elles-mêmes des dépendances.  Donc à l'exécution du code, je reçois immédiatement ce genre d'erreur.

```
<terminated> HBaseConnector [Java Application] C:\Program Files\Java\jre1.8.0_65\bin\javaw.exe (2015-11-16 10:33:23)
Exception in thread "main" java.lang.NoClassDefFoundError: org/apache/commons/logging/LogFactory
        at org.apache.hadoop.conf.Configuration.<clinit>(Configuration.java:169)
        at HBaseConnector.main(HBaseConnector.java:19)
Caused by: java.lang.ClassNotFoundException: org.apache.commons.logging.LogFactory
        at java.net.URLClassLoader.findClass(Unknown Source)
        at java.lang.ClassLoader.loadClass(Unknown Source)
        at sun.misc.Launcher$AppClassLoader.loadClass(Unknown Source)
        at java.lang.ClassLoader.loadClass(Unknown Source)
        ... 2 more
```

Figure 30 - capture d'écran - première série d'erreurs

Il est évident qu'il me manque des librairies. Je peux utiliser un outil de dépistage des dépendances de libraires, mais pour cet exercice, je préfère le faire à la main. Les librairies nécessaires pour rouler ce programme sont :

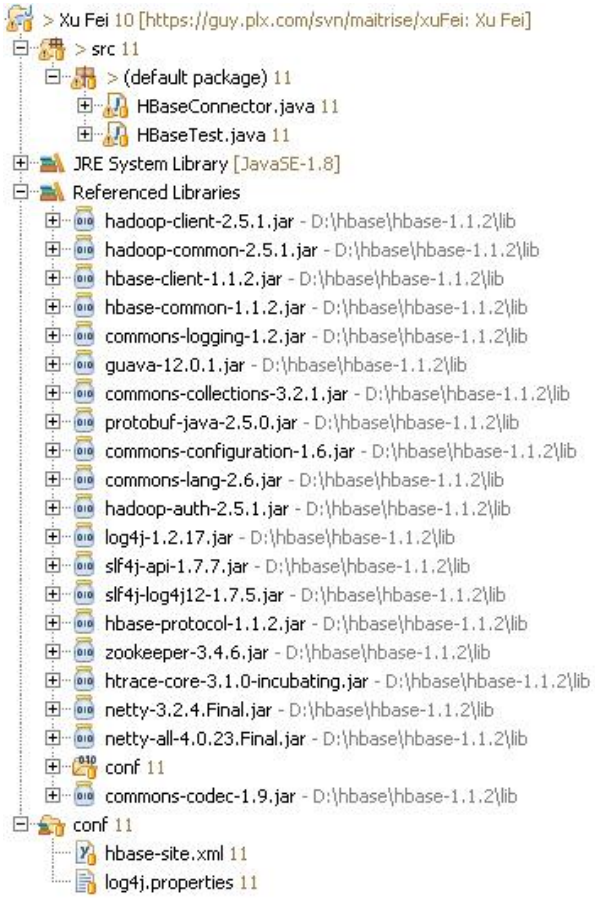


Figure 31 - capture d'écran - la liste des librairies

Avant d'exécuter le programme de l'exemple #1, il faut créer une table dans HBase en utilisant le « hbase shell »:

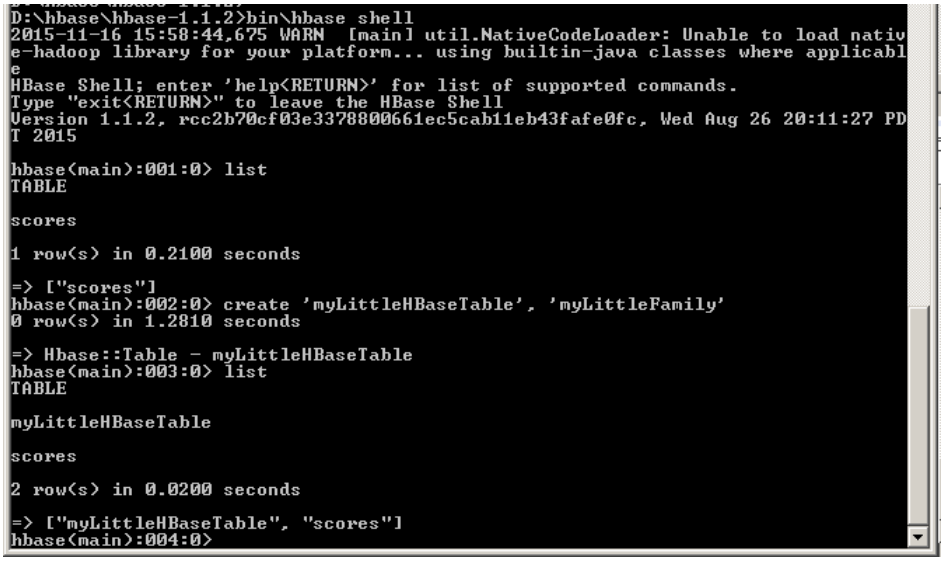


Figure 32 - capture d'écran - création d'une BD dans HBase

Il a été aussi nécessaire de créer un fichier « hbase-site-xml » pour fournir les informations nécessaires pour la connexion à notre instance HBase locale :

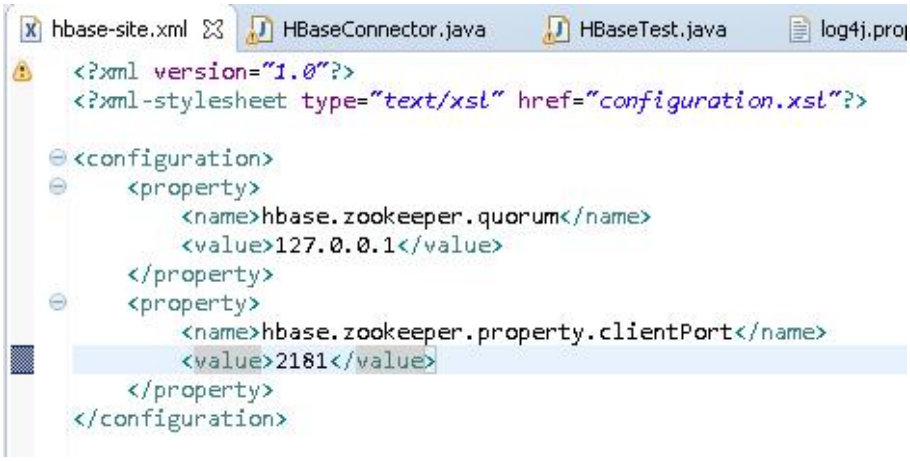


Figure 33 - capture d'écran - hbase-site.xml

Il est à noter que même si j'ai réussi dans ma configuration du hbase-site.xml, j'ai encore beaucoup de questions au sujet des différentes options de configuration des différentes

composantes de ce genre d'environnement. C'est un projet pour le futur. Et l'exécution du programme est un succès :


Figure 34 - capture d'écran - programme avec succès

### 7.5.4 Exemple #2 dans le blogue de Xu Fei

Le deuxième exemple (annexe B) n'inclut pas autant de commentaires, mais est écrit de façon à être très lisible et auto documentée.

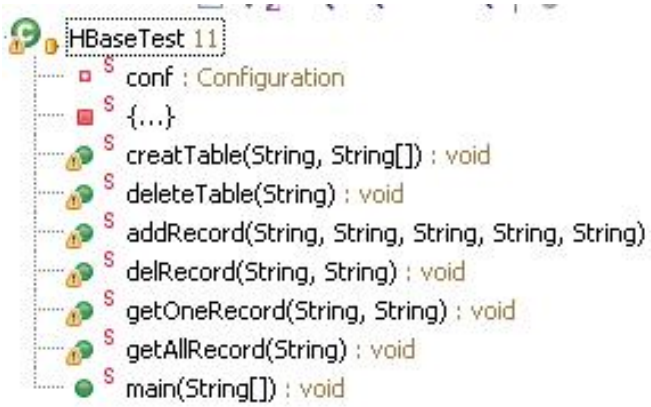


Figure 35 - capture d'écran - classes dans exemple #2

Le tout se fait de façon beaucoup plus rapide, car tous les prérequis sont déjà en place. Exécution avec succès.

```
table already exists!
add records
insert recored zkb to table scores ok.
insert recored zkb to table scores ok.
insert recored zkb to table scores ok.
insert recored zkb to table scores ok.
insert recored baoniu to table scores ok.
insert recored baoniu to table scores ok.
===========get one record========
zkb course: 1447708803300 90
zkb course:art 1447708803305 87
zkb course:math 1447708803300 97
zkb grade: 1447708803290 5
===========show all record========
baoniu course:math 1447708803310 89
baoniu grade: 1447708803305 4
zkb course: 1447708803300 90
zkb course:art 1447708803305 87
zkb course:math 1447708803300 97
zkb grade: 1447708803290 5
===========del one record========
del recored baoniu ok.
zkb course: 1447708803300 90
zkb course:art 1447708803305 87
zkb course:math 1447708803300 97
zkb grade: 1447708803290 5
===========show all record========
zkb course: 1447708803300 90
zkb course:art 1447708803305 87
zkb course:math 1447708803300 97
zkb grade: 1447708803290 5
```

Figure 36 - capture d'écran - exemple #2 succès

### 7.5.5 MapReduce Tutorial – Hadoop.apache.org

À cette étape, j'ai besoin de comprendre un peu plus sur MapReduce. Je suis le tutoriel que je retrouve sur le site web d'Apache (The Apache Software Foundation, s.d.). Les vidéos de Lynn Langit que j'ai visionné sur YouTube ont été très instructives sur le fonctionnement de MapReduce.

Maintenant, je vais suivre le tutoriel d'Apache et utilisez le programme « wordcount.java » que l'on peut retrouver sur la page du tutoriel. C'est un succès!

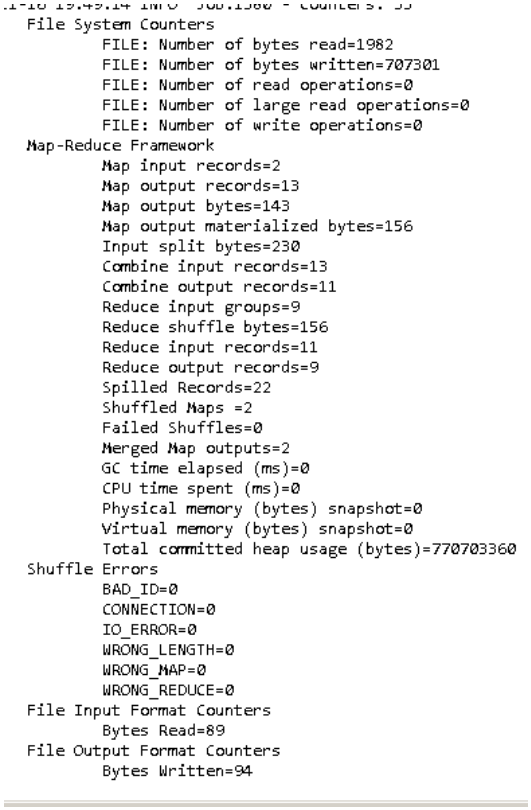


Figure 37 - capture d'écran - map reduce succès

MapReduce me donne mon fichier « _SUCCESS » et mon résultat dans le fichier « part-00000 »

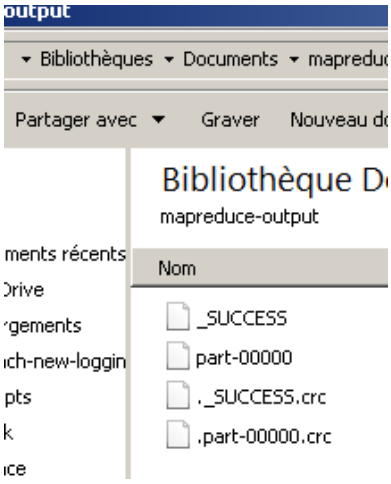


Figure 38 - capture d'écran - MapReduce donne fichier SUCCESS

Voici une capture d'écran pour les 2 fichiers d'entrées (fichier1.txt/fichier2.txt avec jeux minuscule/majuscule) avec le contenu du fichier « part-00000 » à gauche.

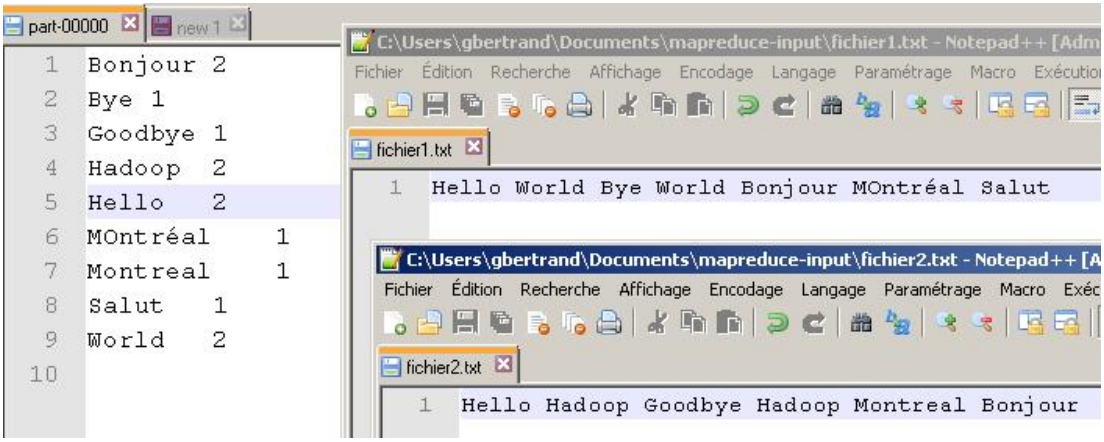


Figure 39 - capture d'écran - fichiers pour MapReduce

Le tutoriel présente ensuite les interfaces de classes Java, qui incluent Mapper, Reducer, JobConf, JobClient, Partitioner, OutputCollector, Reporter, InputFormat, OutputFormat, OutputCommitter et autres. Je mentionne encore les vidéos de Lynn Langit.

Ce tutoriel nous présente ensuite de l'information sur l'environnement d'exécution et les paramètres que nous pouvons modifier pour améliorer la performance des lots de MapReduce. Ce sont des sujets avancés. Et évidemment, la question de la gestion de mémoire est discutée, car c'est un environnement Java. Ce genre d'environnement a toujours eu une sensibilité à la configuration et à la gestion de la mémoire. Le tutoriel présente une version 2.0 de l'application « wordcount », qui exploite certaines des fonctionnalités plus avancées de MapReduce. Un sujet pour étude futur.

# CHAPITRE 8

## Section 3 – Application de technique de rétro-ingénierie dans le contexte de technologies BigData

### 8.1 Introduction

Dans cette troisième partie de mon rapport, l'objectif est la rétro-ingénierie de l'application Oryx, un logiciel qui fait partie d'une solution de système de gestion des processus d'affaires, pour modifier le niveau de persistance (la base de données) de PostgreSQL vers Hadoop.
Tout ceci en utilisant l'approche « Object-Oriented Reengineering Patterns (OORP). Voir mon cadre de Basili à l'annexe III.

Les activités que je veux accomplir sont :

- journal des activités de l'approche OORP,
- ébauche d'un document d'architecture selon ISO 42010,
- ébauche d'un document « notes de mise à jour ».

Le résultat et la mesure de succès seront la documentation des résultats des essais et problématiques.

### 8.2 « Object-Oriented Reengineering Patterns – OORP »

Un de mes objectifs primaires pour ce projet est d'approfondir mes connaissances et compétences dans la rétro-ingénierie d'application. J'ai eu connaissance de ce livre et j'ai demandé pour l'inclure dans mon projet. Avant de continuer, je crois qu'il est opportun de faire un survol de ce livre avant de l'utiliser dans mon projet. Nous connaissons le concept des patrons de conception dans la programmation orientée objet. Les auteurs Demeyer, Ducasse et Nierstrasz (Demeyer, Ducasse, & Nierstrasz, 2002) nous présentent des patrons à utiliser lors d'un projet de réingénierie d'un système informatique.

Ceux-ci se sont basés sur leurs expériences vécues lors de leurs participations dans le projet FAMOOS. Ce projet avait comme objectif d'aider Nokia et Daimler-Benz à donner un deuxième souffle à des applications patrimoniales, qui pourtant étaient récentes et orientées objet. Ces applications souffraient des problèmes typiques de systèmes patrimoniaux. Ceci inclut la problématique de modification de ces systèmes pour suivre les changements dans les exigences des usagers.

Il a été intéressant de revoir les mêmes thèmes que l'on peut retrouver dans la revue de la littérature de ce projet, tel que :

- Le défi de comprendre ce que fait le système. Il faut ensuite comprendre ce qui ne fonctionne pas, s'il y a quelque chose qui ne fonctionne pas,
- Les changements des exigences ont causé une dérive de la conception et de l'architecture de l'application, ce qui rend les modifications ou améliorations presque impossibles. Ceci a été décrit par la loi de Lehman (M.M. Lehman, 1997) sur l'évolution de logiciel.

Avec tous les travaux qu'ils ont effectués sur tous les différents systèmes, qui n'avaient pas nécessairement le même problème, les trois auteurs ont réalisé qu'il y avait répétition de plusieurs de leurs efforts et activités. Un (ou plusieurs) patron est né.

## 8.3 Le cycle de vie de la rétro-ingénierie

Les auteurs nous offrent un survol de la rétro-ingénierie, ses raisons d'existence et les problèmes à résoudre et nous présente le cycle de vie de la rétro-ingénierie. J'ai déjà discuté de ce sujet dans la revue de la littérature, mais j'inclus le diagramme ci-dessous pour référence.

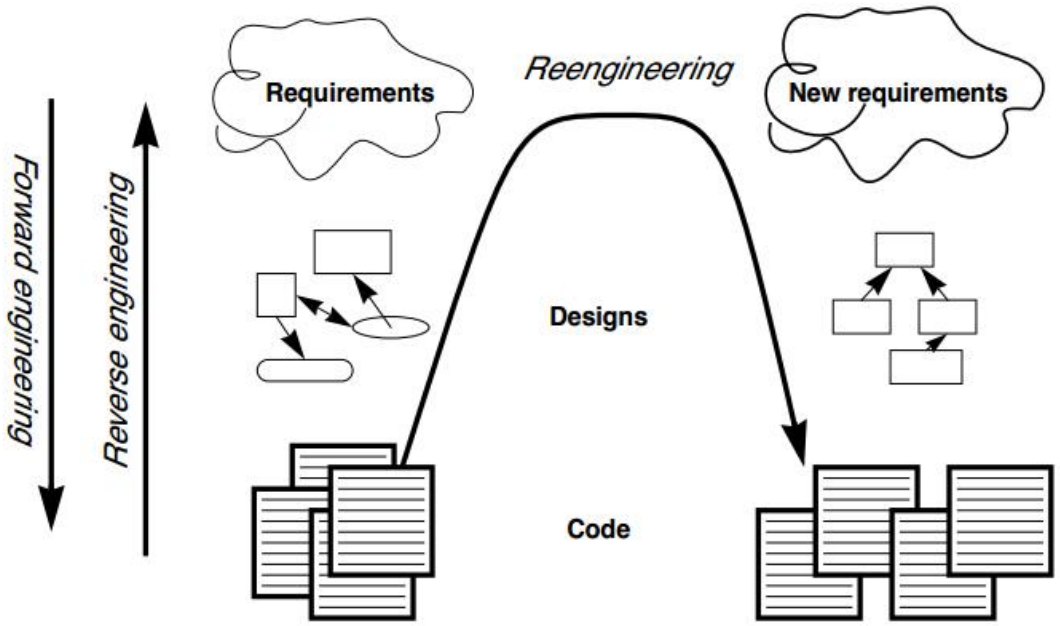


Figure 1.1: Forward, reverse and reengineering

Figure 1 - le cycle de vie de la rétro-ingénierie (Demeyer, Ducasse, & Nierstrasz, 2002)

Les auteurs suivent ce chapitre avec la présentation d'un exemple d'un patron de rétro-ingénierie qui sera utilisé dans le livre. De plus, ils indiquent que les patrons seront présentés selon le cycle de vie de la rétro-ingénierie présenté ci-haut, et qu'ils seront groupés ensemble lorsque nécessaires.

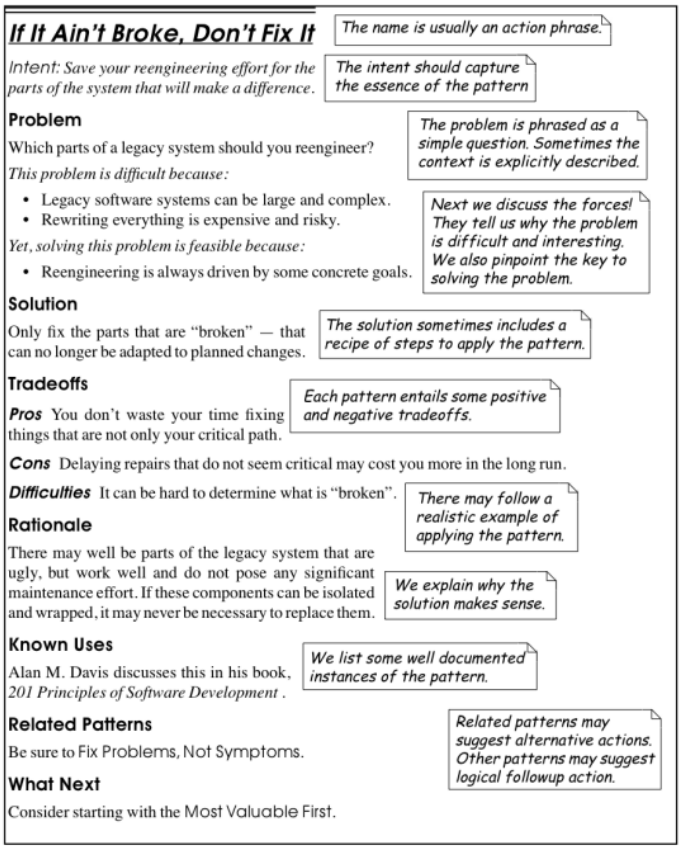


Figure 40 - Exemple d'un patron de rétro-ingénierie (Demeyer, Ducasse, & Nierstrasz, 2002)

Les regroupements des patrons dans le cycle de vie de la rétro-ingénierie.

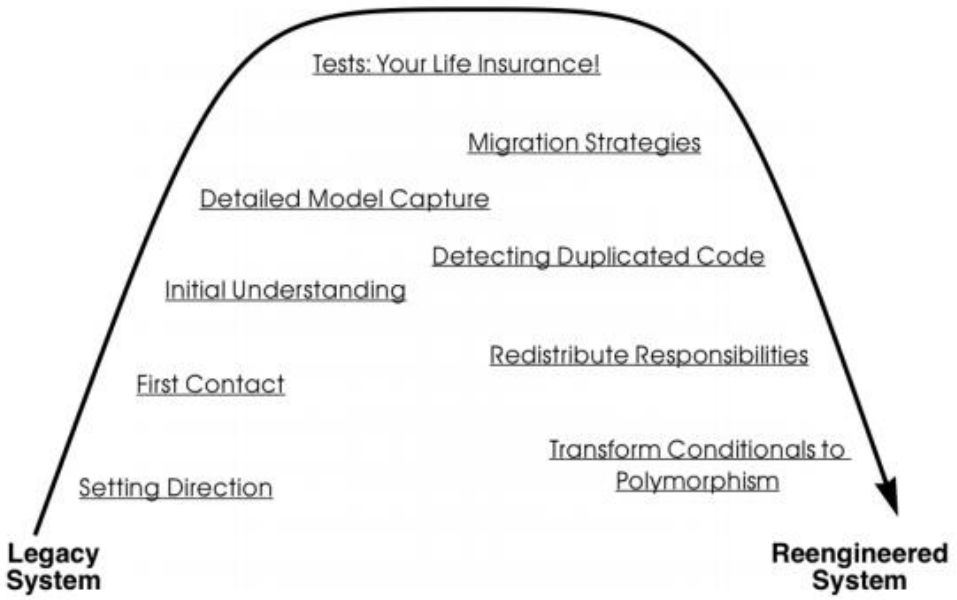


Figure 1.3: A map of reengineering pattern clusters

Figure 41 - patron organisé en groupe dans le cycle de vie (Demeyer, Ducasse, & Nierstrasz, 2002)

Le reste du bouquin décrit les différents patrons. J'utiliserai ceux-ci pour mes travaux sur le logiciel Oryx dans le prochain chapitre.

## 8.4 Le projet Oryx

Le logiciel Oryx, aussi connu comme Projet Oryx est un outil visuel et graphique avec une interface web pour définir des modèles de processus d'affaires. Ces modèles peuvent être ensuite utilisés dans un engin d'exécution, aussi connue comme « moteur de workflow ». C'est un logiciel de source libre dont les créateurs sont :

Oryx is a project of the Business Process Technology research group at the Hasso Plattner Institute of IT Systems Engineering at the University of Potsdam. The research group is led by Professor Mathias Weske, as is the Oryx project.

Figure 42 - Oryx définition (Oryx-Editor, s.d.)

Les usagers utilisent un fureteur pour utiliser le logiciel et créer/modifié leurs modèles. Ils peuvent sauvegarder leurs modèles pour modification ultérieure, ce qui est évidemment fait sur le serveur.

## 8.5 Le problème avec Oryx

Et ceci est le problème à résoudre. Il y a un goulot d'étranglement lorsqu'il y a lecture ou sauvegarde d'un modèle.

## 8.6 Objectif de modification pour Oryx

Donc l'objectif de mon projet est de modifier le système de gestion de base de données utilisé, qui est actuellement PostgreSQL. Le nouveau système de base de données sera HBase, un système de gestion de base de données non relationnelle distribuée.

## 8.7 OORP avec Oryx

### 8.7.1 Groupement patrons : Établir la direction « setting direction »

**Sommaire du groupement du patron** : Dans tout genre de projet, incluant la rétro-ingénierie, il y a toujours une foule de parties prenantes, objectifs variés et autres facteurs qui tentent d'influencer la direction du projet. De plus, il est naturel pour nous de préféré travaillé sur quelque chose qui est intéressant ou possiblement facile à faire. Nous avons besoin d'une direction précise pour avancer dans notre projet, il faut bien la communiquer à notre équipe et il est toujours préférable d'obtenir le consensus de celle-ci.

Le premier groupement de patrons nommés « Setting Direction » ou « établir la direction » inclus les patrons suivantes :

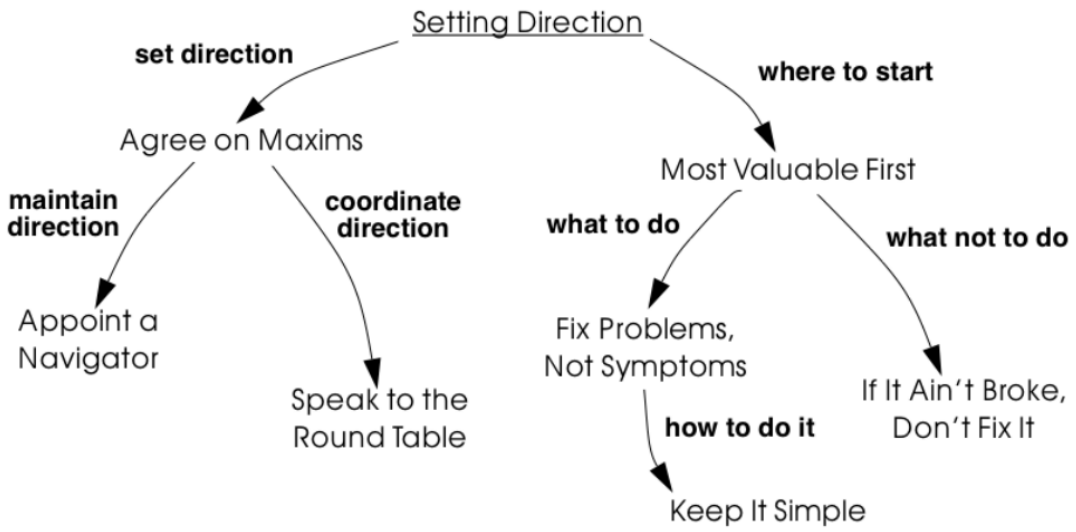


Figure 43 - OORP Groupement patrons établir direction (Demeyer, Ducasse, & Nierstrasz, 2002)

NOTE : Il est évident que ce groupement n'est pas nécessaire dans ce contexte (équipe d'une personne pour ce projet), mais je l'inclus aux fins de documentation.

#### 8.7.1.1 Établir un accord de principe et règle de conduite (Agree on Maxims)

**Sommaire du patron**: Il est important de s'assurer que l'équipe travaille dans la même direction et pour le même objectif. Il est très facile pour un projet de déraper. Ceci aide l'équipe à bien comprendre l'objectif du projet. Cette étape aide aussi à encourager la contribution des participants au succès du projet.

**Résultat Oryx** : Non applicable. Comme je suis une équipe d'une personne pour mon projet, ce patron n'est pas nécessaire.

#### 8.7.1.2 Nommé un navigateur (Appoint a Navigator)

**Sommaire du patron**: Je dois noter que ce patron demande que l'on nomme un navigateur. Ceci n'est pas la même chose qu'un directeur ou chargé de projet. La responsabilité du navigateur est de s'assurer que la vision du projet est respectée.

**Résultat Oryx** : Non applicable. Comme je suis une équipe d'une personne pour mon projet, ce patron n'est pas nécessaire.

#### 8.7.1.3 La communication dans l'équipe (Speak to the round table)

**Sommaire du patron**: Une équipe de rétro-ingénierie fait face à un projet d'archéologie et doit découvrir des faits typiquement cachés dans une application. Les connaissances acquises lors de cette courbe d'apprentissage doivent être communiquées avec les autres membres de l'équipe de façon régulière pour en extraire le maximum de bénéfice et pour s'assurer du succès du projet.

**Résultat Oryx** : Non applicable. Comme je suis une équipe d'une personne pour mon projet, ce patron n'est pas nécessaire.

#### 8.7.1.4 Établir la priorité des problèmes (Most Valuable First)

**Sommaire du patron**: Un système patrimonial peut avoir une quantité surprenante de problèmes. Il faut savoir choisir les priorités.

**Résultat Oryx** : Non applicable. Pour ce projet, il n'y a qu'un problème à résoudre.

#### 8.7.1.5 Correction des problèmes et non des symptômes (Fix Problems, Not Symptoms)

**Sommaire du patron**: Il faut s'attaquer à la source des problèmes et pas nécessairement aux symptômes énoncés par les usagers ou les parties prenantes.

**Résultat Oryx** : Même si l'objectif du projet est de corriger le goulot d'étranglement lors de la lecture et l'écriture dans le système de base de données, il n'y a aucun fait connu qui démontre que c'est la source du problème. Cependant, un des objectifs de mon projet est l'apprentissage de la technologie BigData, donc c'est par défaut le problème à corriger.

#### 8.7.1.6 Corriger que les vrais problèmes (If it ain't broke, don't fix it)

**Sommaire du patron**: Il faut prendre soin de corriger ou modifier que les parties nécessaires dans un système patrimonial.

**Résultat Oryx** : Pour l'objectif de ce projet, le problème et la correction est la migration de PostgreSQL vers HBase.

#### 8.7.1.7 La simplicité (Keep it simple)

**Sommaire du patron**: Lors du processus de réingénierie, nous avons tendance à ajouter de la flexibilité et simplicité. Il est fort possible d'y introduire trop de flexibilité et créer une application qui est encore plus difficile de modifier celui-ci dans le futur.

**Résultat Oryx** : Ce patron est intéressant, car lors de ma planification, je pensais justement d'ajouter des paramètres à la configuration pour permettre à l'administrateur de choisir d'utiliser la SGBD PostgreSQL ou HBase, et possiblement d'ajouter un autre paramètre pour activer l'effet de migration (lecture PostgreSQL – écriture HBase).
La simplicité dicte que je ne ferai aucune de ces options.

### 8.7.2 Groupement de patrons : Première aperçue (First Contact)

**Sommaire du groupement de patrons** : Ce deuxième groupement de patrons décrit les étapes à accomplir lors de notre premier accès au logiciel/système. La question est toujours : où faut-il commencer? Sauf le manque de connaissance sur le logiciel, le plus gros problème est toujours la demande d'une estimation des travaux à accomplir. Une évaluation initiale est toujours difficile et les décisions qui seront prises à ce stage peuvent avoir des conséquences non désirées dans le futur. Il est aussi possible que ces décisions soient complètement erronées. L'objectif des patrons dans ce groupement est de résoudre le conflit parmi les items suivants :

- Un système patrimonial est typiquement énorme et complexe. Une équipe de rétro-ingénierie doit connaitre ses capacités et décider si elle est capable d'entreprendre un projet de cette grosseur. Il est possible que l'équipe doive décider de diviser le projet en plus petit morceau,
- Le laps de temps est très court. Au début du projet, il faut utiliser notre temps pour évaluer les mérites du projet avec l'information qui nous est disponible. Il est facile de tomber dans le piège de perte de temps dans certaines activités faciles ou familières. Il est important de s'attaquer au cœur du problème,
- Nos premières impressions et perceptions sont possiblement dangereuses et incorrectes. Il ne faut pas prendre une décision précipitée avec de l'information incomplète,
- Il faut comprendre les objectifs parfois cachés des autres membres de l'équipe de réingénierie.

Le plus gros risque lors du premier contact avec un système inconnu est la perte de temps. Ce regroupement de patrons devrait être utilisé pendant un court laps de temps, possiblement lors de la première semaine du projet. À la fin de la semaine, il devrait y avoir une connaissance des problèmes majeurs pour commencer la planification du projet, ou au besoin annuler le projet.

À la fin de cette étape, il est possible de créer le premier plan de projet.

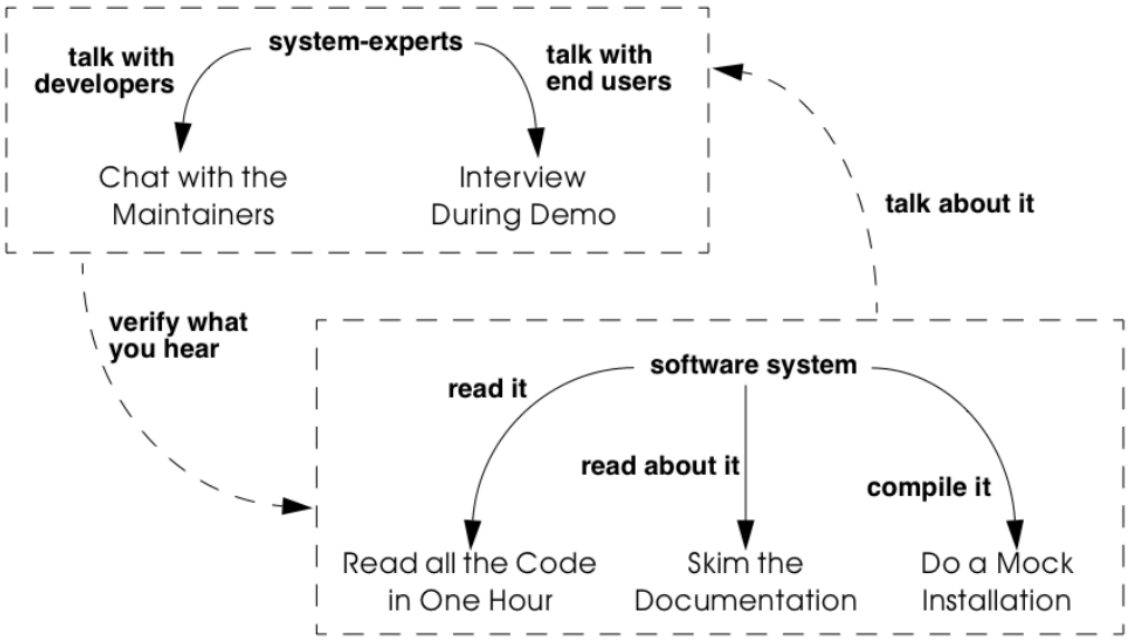


Figure 44 - Groupement patrons "Première aperçue" (Demeyer, Ducasse, & Nierstrasz, 2002)

#### 8.7.2.1 Discussion avec l'équipe de maintenance (Chat with the mainteners)

**Sommaire du patron**: Je note que les auteurs ont dédié une grande quantité de temps sur cette section. Ils mentionnent avec enthousiasme que cette étape peut fournir une quantité importante d'information et aussi peut aider à la création d'engagements envers le projet par tous ses « nouveaux » membres de diverse organisation qui y participe.

**Résultat Oryx** : Non applicable.

#### 8.7.2.2 Faire une révision du code en une heure (Read all the code in one Hour)

**Sommaire du patron**: Il est important de vérifier le code pour évaluer son état actuel et surtout sa qualité. Après une heure, il devrait être possible :

- D’évaluer la possibilité de faire ce projet de rétro-ingénierie,
- D’identifier les composantes majeures du système,
- D’identifier rapidement le niveau de compétence et maturité du codage,
- Identification des recherches subséquentes nécessaires.

**Résultat Oryx** : Le logiciel Oryx a une architecture polyglotte. L’application contient du Java, JavaScript, Python. L’application typiquement roule sur un site web (Tomcat), mais elle peut aussi être convertie en application Windows à l’aide de XULRunner de Mozilla. Il y a aussi du CSS, XSLT, XSD et HTML, mais ceux-ci peuvent être considérés comme normaux pour une application Web. Je trouve les librairies Log4J et SLF4J qui sont inclus dans le projet. Ces librairies sont utilisées pour faire de la journalisation des activités du code lors de son exécution. Cependant, ma lecture rapide du code indique une très petite utilisation de cette fonctionnalité. Je trouve aussi que la section serveur utilise Hibernate pour la persistance, dans un système de base de données PostgreSQL. Les librairies Hibernate sont utilisées dans une application pour stocker nos données au niveau de la persistance avec un modèle orienté objet. Une application qui utilise Hibernate n’a plus besoin de faire l’utilisation d’appel direct dans la base de données, mais utilise plutôt les appels Hibernate. Ce point est important et j’en discuterai dans une autre section.

De plus, il semble que plusieurs des contributeurs étaient de pays ou de langue maternelle non anglophone, donc certains commentaires, nom de classes ou méthode et même les acronymes sont plus difficiles à comprendre. On remarque aussi qu'il y a de différentes contributions. Il existe des « packages » Java d’une hiérarchie complètement différente. (exemple : org.b3mn.poem vs de.hpi.bpt)

Cependant, mon objectif se limite à la migration de PostgreSQL à Hbase. Une recherche du code source pour le mot « hibernate » ne me donne que 7 fichiers. Un coup d’œil rapide sur ces 7 fichiers me donne l’espoir que la modification sera plus facile que prévu.

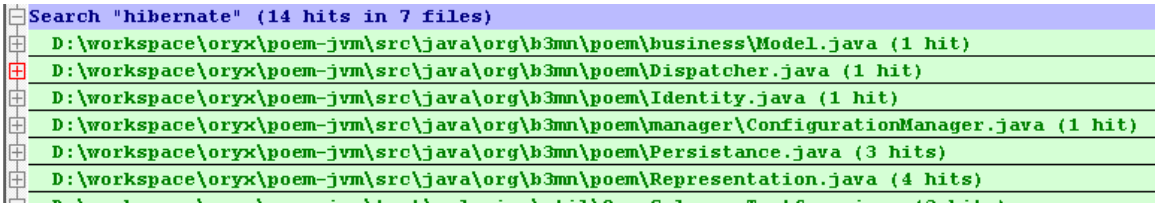


Figure 45 - capture d'écran - identification des classes Java qui utilisent Hibernate

Je note aussi que ces classes Java n'utilisent pas la journalisation Log4J et SLF4J que j'ai mentionnée plus haut. Aucune journalisation lors de l'exécution du code pour les classes qui m'intéresse.

Les auteurs de l'OORP suggèrent de faire une installation de test pour évaluer le potentiel pour la rétro-ingénierie.

Ma première tentative a été de tenter de faire rouler le client autonome ( thick client ). Dans le dossier « standalone », il y a un fichier « README.txt » qui fournit qu'une instruction : télécharger « xulrunner » ici et ensuite exécuter le fichier « run.bat ». Le résultat a été rapide et surprenant :

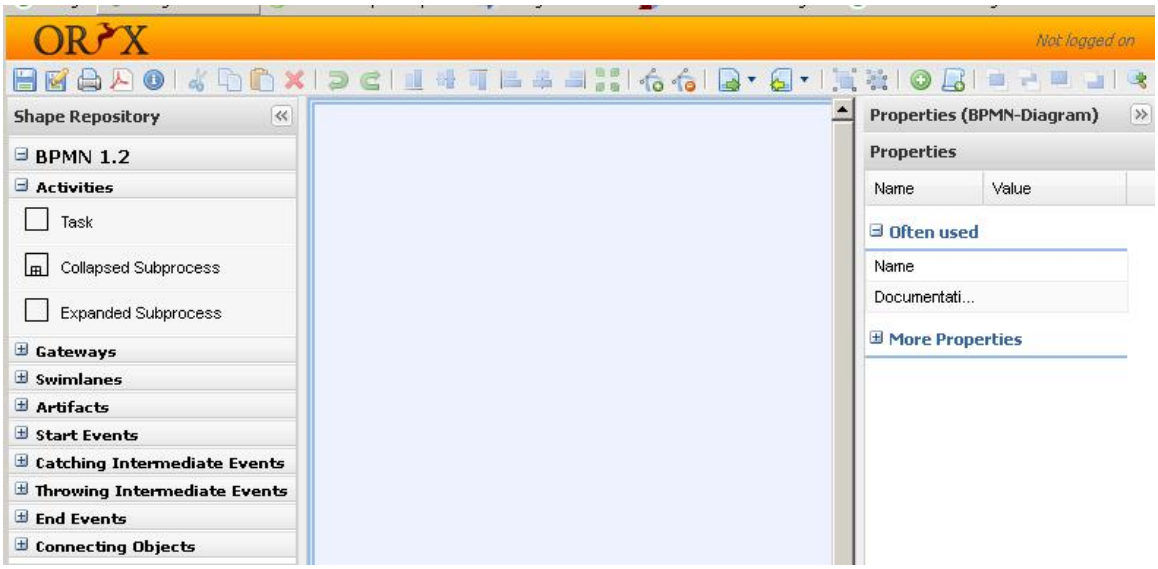


Figure 46 - capture d'écran - le client autonome "thick client"

Cependant, il n'y a pas d'instruction pour modifier le fichier « application.ini » qui de toute évidence, devrait contenir les informations pour configurer le serveur avec lequel l'application doit communiquer. Je trouve aussi des fichiers « war » qui fonctionnent avec Tomcat. Je les installe et ceci me donne une bonne idée du processus d'installation et une partie des fichiers sources, surtout les fichiers utilisés pour le « web », tels que les CSS, HTML, XSLT, etc.

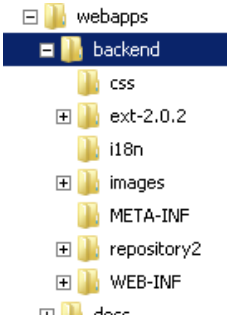


Figure 47 - capture d'écran - structure typique d'une installation Tomcat

#### 8.7.2.3 Faire un survol rapide de la documentation « Skim the Documentation »

**Sommaire du patron**: la documentation pour les usagers ainsi que la documentation de développement peuvent donner un bon indice pour évaluer les efforts requis pour accomplir la rétro-ingénierie.

**Résultat Oryx** : Celle-ci est très facile. Il n'y a aucune documentation qui est fournie avec le code source. Je dois donc me fier à la documentation que je peux trouver sur le site web de « Google Code » (Oryx-Editor, s.d.)

La documentation est mince, mais il existe une question sur la page de l'architecture de la base de données qui me laisse perplexe:


Figure 48 - Question suspecte (Oryx-Editor, s.d.)

Je n'y trouve pas de fichier Python dans le code source. Y a-t-il des fonctions Python dans le schéma de BD? Est-ce des procédures stockées? C'est à vérifier dans une section subséquente. En général, la documentation sur le site web est légère, mais utile.

#### 8.7.2.4 Visionnez une démonstration et posez des questions ( Interview during Demo)

**Sommaire du patron**: une démonstration peut donner des indices sur le fonctionnement du logiciel et aider dans l'évaluation de l'effort requis pour la rétro-ingénierie.

**Résultat Oryx** : non applicable.

#### 8.7.2.5 Faire une installation test (Do a Mock Installation)

**Sommaire du patron**: vérifier que vous avec tous les morceaux nécessaires pour compiler et installer l'application. Possiblement qu'il y a de l'information cachée. Un outil de plus pour évaluer le travail à faire.

**Résultat Oryx** : En réalité, j'ai déjà accompli cette étape dans la section de la lecture rapide du code. Le code binaire de l'application m'a été fourni avec le CD que j'ai reçu. Mais je dois faire une compilation complète. Je me concentre sur la section « backend » et le résultat est positif.

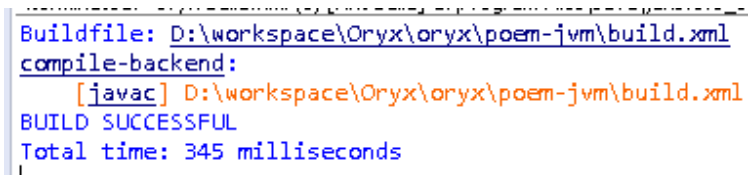


Figure 49 – capture d'écran - compilation avec succès

L'architecture du code, spécifiquement la section « serveur » est conçue pour être utilisé avec Apache Tomcat. Typiquement, les classes Java sont compilées et ensuite stockées dans un fichier WAR. Ce fichier WAR est ensuite copié dans le sous-répertoire WEBAPPS de l'installation Tomcat, et l'application commence à être exécutée.

L'application contient aussi des fichiers de contrôle « build.xml » pour utilisation avec Apache ANT. Une étude rapide démontre que toutes ces fonctionnalités sont incluses dans les fichiers ANT, incluant la compilation, la création des fichiers WAR et même l'installation des fichiers WAR dans Apache Tomcat. Excellent!

### 8.7.3 Groupement patrons : Premier survol de l'application (Initial Understanding)

**Sommaire du groupement du patron** : Dans la section précédente, l'objectif était axé sur la collecte de données préliminaire pour être capable de comprendre les fonctionnalités de l'application.

Maintenant, nous avons besoin d'une meilleure compréhension de l'architecture et le design de l'application. Si notre objectif est d'extraire une fonctionnalité particulière, est-ce qu'il sera possible de faire ceci?

À la fin de ce groupe de patrons, nous devrions avoir assez d'information pour avoir une ébauche des activités de rétro-ingénierie nécessaires.

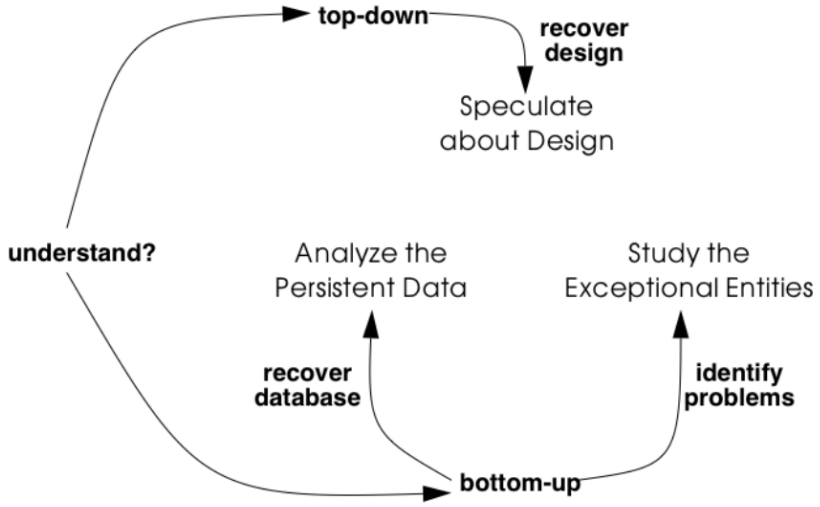


Figure 50 - Groupement patrons - premier survol (Demeyer, Ducasse, & Nierstrasz, 2002)

#### 8.7.3.1 Faire une analyse du niveau de la persistance (Analyse the Persistent Data)

**Sommaire du patron** : comprendre les données qui sont stockées de façon persistance, typiquement dans un système de gestion de base de données.
Mais il faut faire attention. Est-ce que toutes les données stockées dans une base de données sont importantes? Quand la lecture de ces données est faite dans une application, elles sont typiquement transformées dans une structure complexe. Cette transformation est rarement bien documentée.

**Résultat Oryx** : Le site web « Oryx-Editor database schema » nous offre ce schéma de la BD.

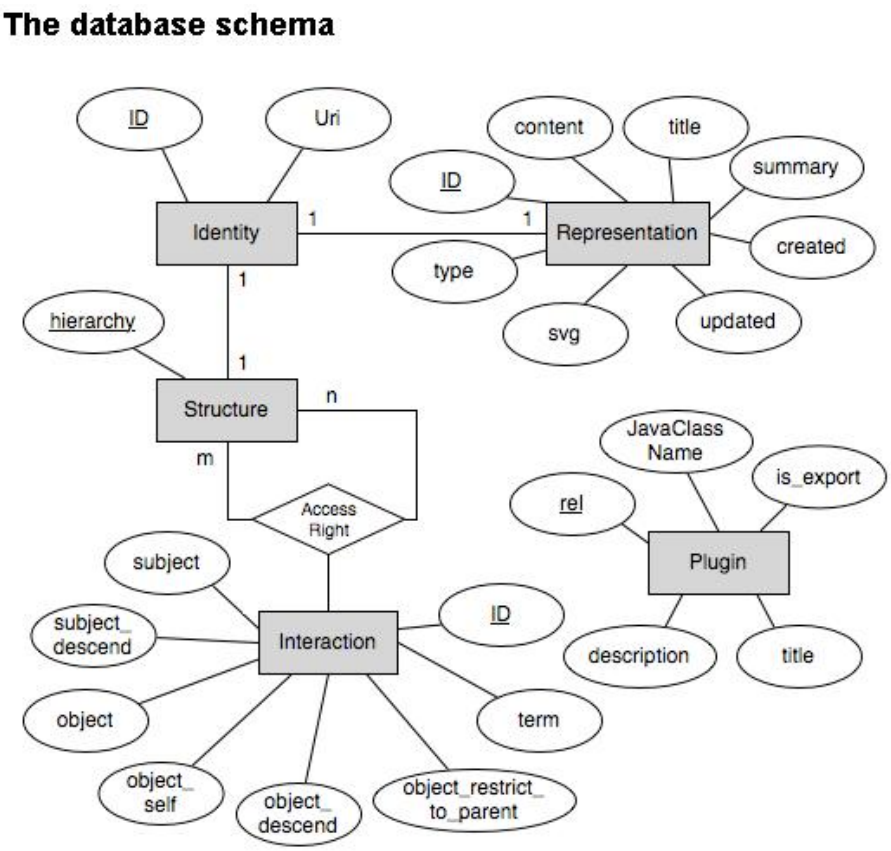


Figure 51 - schéma de la BD selon le site web (Oryx-Editor, s.d.)

Ce diagramme me semble simple…je doute qu'il y ait seulement 5 tables. Je retrouve le script de création du schéma de la base de données 'db_schema.sql'. J'inclus une copie à l'annexe A.

Maintenant, j'installe la base de données. Il faut créer un usager POEM et sa base de données avec le même nom. Ensuite, on roule le script db_schema.sql.

```
C:\Users\gbertrand>createuser -U postgres --echo --pwprompt --encrypted poem
Enter password for new role:
Enter it again:
Shall the new role be a superuser? (y/n) n
Shall the new role be allowed to create databases? (y/n) y
Shall the new role be allowed to create more new roles? (y/n) y
Password:
CREATE ROLE poem ENCRYPTED PASSWORD 'md54fae7683b3e8809f364b0026c885af8c' NOSUPE
RUSER CREATEDB CREATEROLE INHERIT LOGIN;

C:\Users\gbertrand>createdb -U postgres --echo --encoding utf8 --owner poem poem

Password:
CREATE DATABASE poem OWNER poem ENCODING 'utf8';

C:\Users\gbertrand>
```

Figure 52 – capture d'écran - création de l'usager POEM et la BD=Poem

Succès!!! Voir annexe B pour le résultat du script db_schema.sql

J'utilise l'outil d'administration 'pgAdmin III' de PostgreSQL pour voir le résultat :

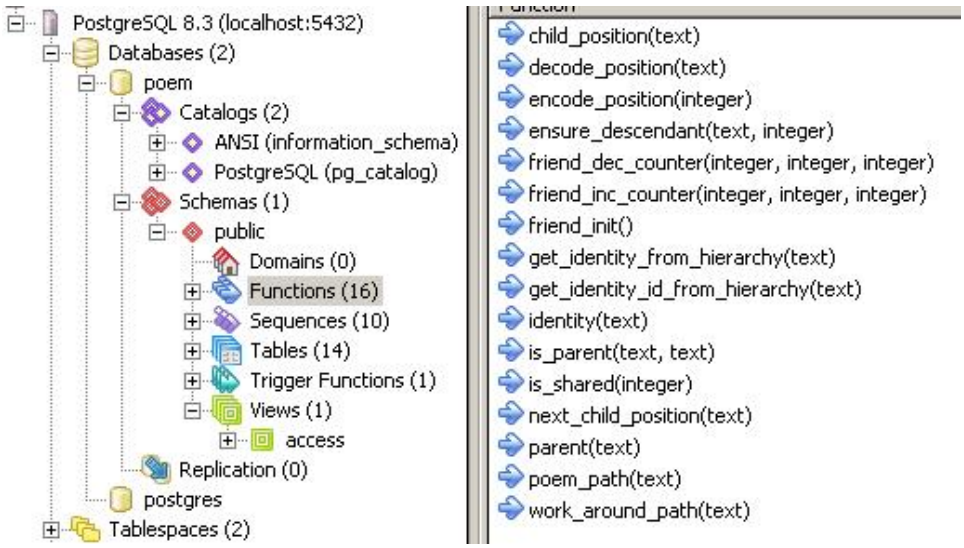


Figure 53 - capture d'écran - les fonctions or procédure stockée

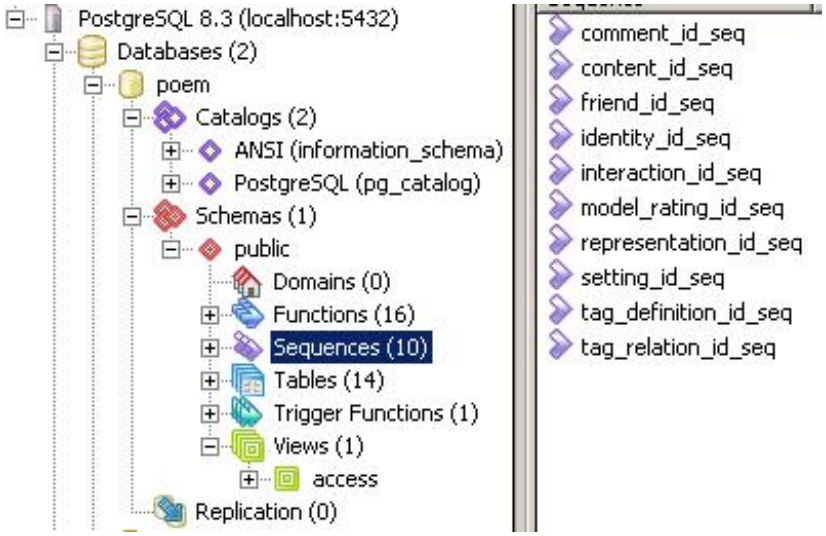


Figure 54 - capture d'écran - les séquences

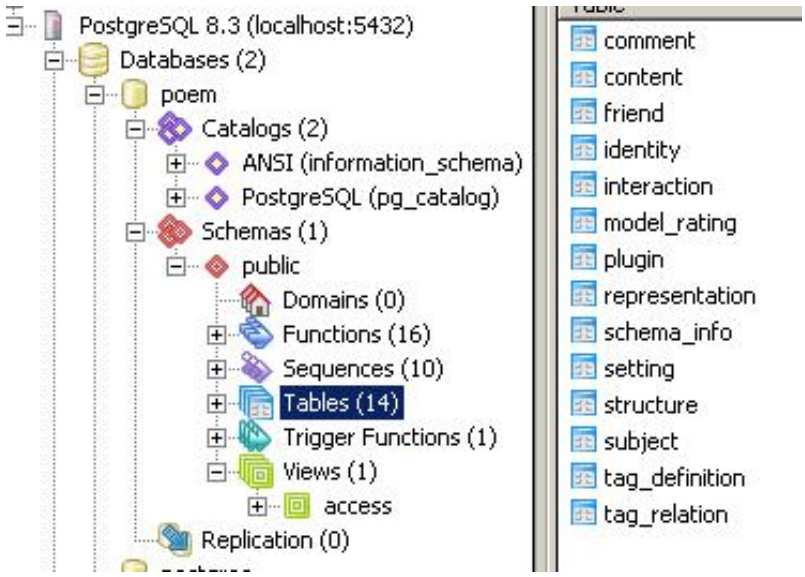


Figure 55 - capture d'écran - les tables

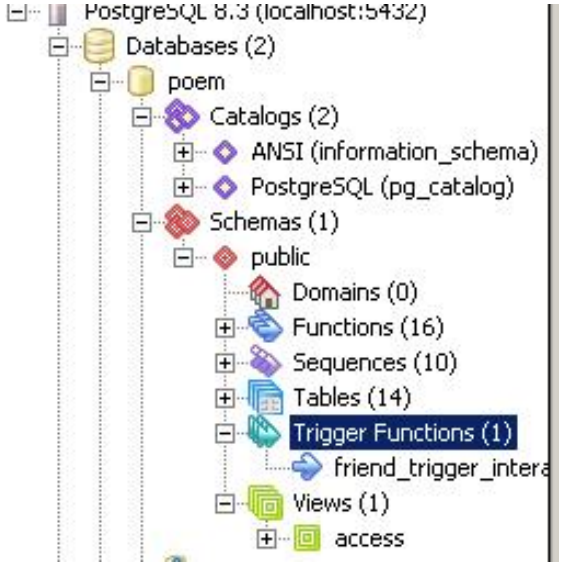


Figure 56 - capture d'écran - et un déclencheur automatique (trigger)

Suivant la création de la BD, je retrouve les items suivants :

- 14 tables,
- 10 séquences,
- 16 fonctions ou procédure stockée, certaines en PL/SQL et d'autres en Python,
- 1 déclencheur.

L'item qui est important pour mon projet est le déclencheur et les fonctions. Ces items sont codés spécifiquement selon les spécifications de PostgreSQL et il est impossible de les utiliser dans d'autres systèmes de gestion de base de données. Une conversion, ou plutôt, une réécriture sera nécessaire.

#### 8.7.3.2 Créer des hypothèses sur le design de l'application (Speculate about Design)

**Sommaire du patron** : À ce stage, on devrait être capable de raffiner notre compréhension du design de l'application. C'est à ce stage que l'on peut commencer à faire des diagrammes d'architecture de haut niveau et possiblement aussi des diagrammes de classes préliminaires.

**Résultat Oryx** : Le code source n'est pas organisé pour être utilisé dans un éditeur intégré comme Eclipse. La structure des répertoires et les autres composantes du code source indiquent que ceux qui ont travaillé sur ce code dans le passé ont possiblement utilisé un éditeur de texte standard. On retrouve plusieurs technologies, des langages utilisés pour le code source et une structure de l'architecture compliquée. C'est une application polyglotte. On retrouve différentes technologies, différentes approches et aussi il semble y avoir des items qui ne sont plus utilisés, mais personne n'a osé les effacer du projet. Exemple les multiples procédures stockées dans la base de données, mais non utilisées.

**NOTE IMPORTANTE**

La découverte de l'utilisation de Hibernate a un impact majeur sur mon projet.

Hibernate offre « Hibernate OGM » pour utilisation avec les bases de données « NoSQL ». Un changement de librairie, une modification au fichier .XML pour la configuration de Hibernate, et c'est terminé. [5] [6]

Je discuterai de l'impact de ceci sur mes objectifs de mon projet dans ma section d'interprétation à la fin de mon projet. Je continue avec les différentes étapes de l'OORP pour ma propre éducation. (Bernard, s.d.) (Hightower, s.d.)

Voir le chapitre 9 pour plus d'information sur Hibernate.

#### 8.7.3.3 Découvrir les problèmes potentiels de design (Study the Exception Entities)

**Sommaire du patron** : Il est important de découvrir s'il y a des problèmes dans le design de l'application, mais à ce stage, nous n'avons pas assez d'informations précises pour séparer les bons et mauvais design dans l'application. C'est le temps d'utiliser des outils d'analyse de code statique pour nous aider.

**Résultat Oryx** : Il est évident que je me concentre sur l'aspect de la persistance. Mais je remarque beaucoup plus d'autres problèmes potentiels avec les autres morceaux de l'application.

On peut retrouver des classes qui semblent faire beaucoup trop de chose, le terme anglais est « god classes ».

On retrouve aussi des valeurs non modifiables directement dans le code source, tel que : « Writer writer=new FileWriter("**C:/Program Files/Apache Software Foundation/Tomcat 6.0/webapps/json/**"+count+".json"); ». Il est préférable d'utiliser un fichier de configuration.

On retrouve aussi ce genre de commentaire:

```
} finally {
    // This might be a hibernate design crime but should fix some problems
    Persistance.getSession().close();
}
```

Figure 57 - capture d'écran - des commentaires dans le code source

Ceci indique qu'il y a possibilité d'une mauvaise utilisation de Hibernate.

### 8.7.4 Groupement de patrons : Acquisition des détails du modèle (Detailed Model Capture)

**Sommaire du groupement de patrons** : À ce stage, nous devrions avoir assez d'information pour créer un modèle des sections de l'application qui seront importantes pour notre projet de rétro-ingénierie. Ce groupement requiert plus de connaissances techniques, avec beaucoup d'utilisation d'outils. Selon les auteurs de l'OORP, nous n'avons pas encore décidé si ce projet de rétro-ingénierie devrait continuer, ou si une réécriture serait plus abordable.

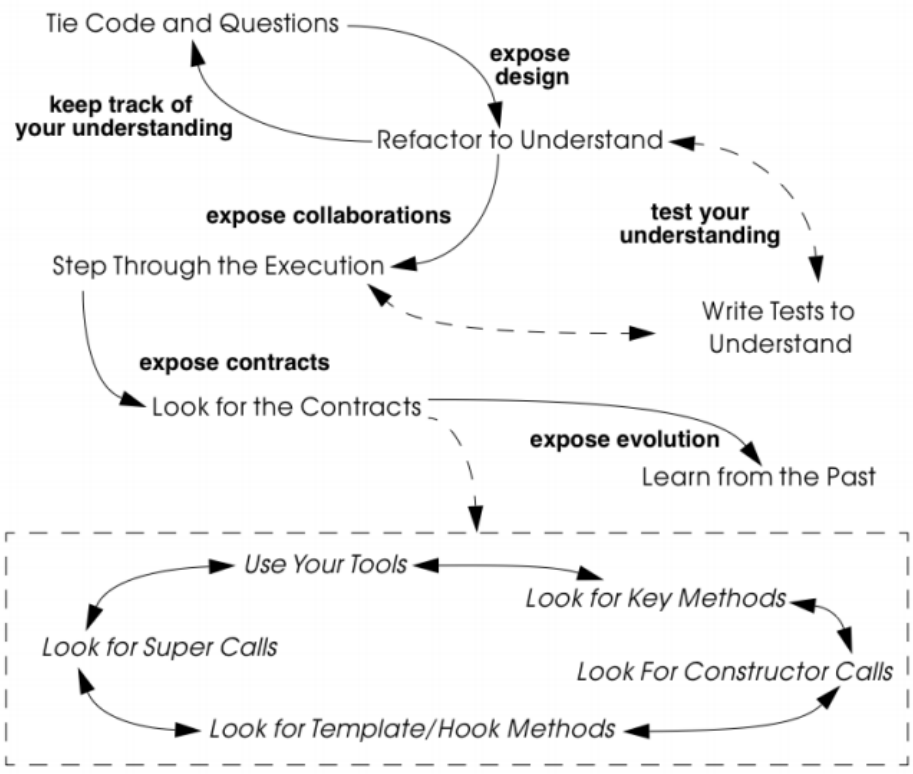


Figure 58 - groupement de patrons - Acquisition des détails du modèle (Demeyer, Ducasse, & Nierstrasz, 2002)

#### 8.7.4.1 Faire le lien entre vos questions et le code (Tie Code and Questions)

**Sommaire du patron** : Il est difficile de synchroniser les liens entre un document que vous créez avec vos questions et réponses et vos commentaires au sujet du code source, et le code source lui-même. Au lieu de créer un document séparé, écrivez vos commentaires directement dans le code source avec une notation unique pour les retrouver rapidement.

**Résultat Oryx** : Pour le code Java, les librairies Log4J et SLF4j sont déjà dans le projet, mais je remarque qu'ils ne sont pas utilisés dans les 7 classes Java que j'avais identifiées dans une section précédente. En ce qui concerne les procédures PL/SQL et Python stockés dans la BD, je peux utiliser la commande « RAISE NOTICE ».

Un exemple d'ajout d'entré de journal. La méthode « newModel » dans la classe « Identity.java » n'a pas de journalisation. En ajoutant l'instruction « logger.debug… », nous aurons de l'information pertinente. Celle-ci nous permettra de suivre le déroulement du logiciel.

```
private static Logger logger=Logger.getLogger(Identity.class);
...
...
public static Identity newModel(Identity owner, String title, String type,
String summary, String svg, String content)
{
logger.debug("creating new model: owner ID=" + owner.id + ",owner=" + owner.uri
+  ",title=" + title + ",type=" + type + ",summary=" + summary + ",svg=" +
svg);
```

#### 8.7.4.2 Réingénierie logicielle pour comprendre (Refactor to Understand)

**Sommaire du patron** : Sans faire de modification de logique dans le code, faire des modifications d'une section du code pour valider votre compréhension de celle-ci. Assurez-vous d'avoir des tests pour valider le résultat.

**Résultat Oryx** : Ce patron sera utilisé que si l'on retrouve une classe du genre je-fais-tout, aussi nommé « god class » en anglais. Après une étude des journaux qui seront créés dans le patron précédent, une décision sera prise à ce sujet.

#### 8.7.4.3 Naviguer à travers l'exécution (Step throught the Execution)

**Sommaire du patron** : faites exécuter le code dans un débogueur et étudier la relation entre les objets.

**Résultat Oryx** : L'usage d'un déboguer n'est pas prévue pour l'instant. Les journaux créés par Log4J seront utilisés pour l'instant. Cependant, il est possible d'inclure un serveur Tomcat dans l'exécution d'Eclipse. Ceci permet l'utilisation immédiate du débogueur d'Eclipse.

#### 8.7.4.4 Trouver les contrats (Look for the Contracts)

**Sommaire du patron** : découvrir les contrats entre les classes et la bonne utilisation de ceux-ci

**Résultat Oryx** : Les auteurs du OORP mentionnent que ce patron est difficile. Nous retenons deux commentaires qui démontrent ceci :

- Vous faites confiance que les classes sont utilisées de façon correcte et comme prévue.
- Les contrats sont implicites et il devrait y avoir des indices des relations entre les classes dans le code.

À ce stage, nous ne faisons pas confiance et nous cherchons les indices. Nous sommes encouragés que nous avons seulement 6 classes à vérifier. Nous croyons que les contrats sont déjà connus. Cependant, les classes du genre « god » seront plus difficiles.

#### 8.7.4.5 Apprendre de l'historique (Learn from the past)

**Sommaire du patron** : Est-ce l'historique de l'application donne des indices sur les raisons de l'état de l'application?

**Résultat Oryx** : Une recherche sur Oryx-Editor démontre clairement que cet environnement a été sujet à de multiples modifications par différents établissements éducationnels et thèse d'étudiants au baccalauréat [7].

Le gestionnaire de code source indique qu'il y a eu 3957 modifications de 2007 à 2012. La version disponible sur l'internet est en réalité une copie de la version 1536 du système de code source.

Une étude de ces versions indique que la structure actuelle existe depuis l'origine de l'application et qu'il y a eu peu de changement d'architecture depuis le début, possiblement depuis le milieu des années 2000.

### 8.7.5 Exécution de la rétro-ingénierie

Pour faire suite à nos efforts de compréhension de l'application, nous pouvons conclure que la décision a été prise de continuer avec notre projet de rétro-ingénierie. Nous sommes maintenant dans la section III des patrons de l'OORP, la réingénierie.

#### 8.7.5.1 Les tests : absolument nécessaire (tests: your life insurance)

Le premier groupement de patrons dans cette section suggère d'utiliser des tests. Le mot suggestion est imprécis. Il faut utiliser des tests.

Il est naïf de croire que nous pouvons comprendre de façon précise le fonctionnement d'une section de code source que nous venons d'étudier. Il est encore plus naïf de croire que nous pouvons effectuer une modification et garantir qu'il n'y aura pas d'autres imprévus ou effets secondaires. Il y a simplement trop de risque qu'une modification puisse causer d'autres problèmes.

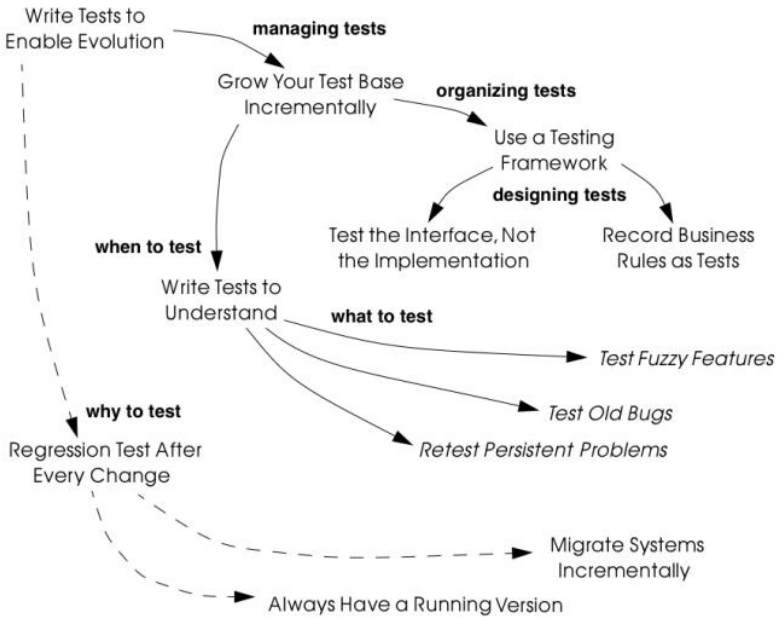


Figure 6.1: When, why, how and what to test.

Figure 59 - Regroupement patrons - les tests

#### 8.7.5.2 Créer des tests pour permettre l'évolution (Write Tests to Enable Evolution)

**Sommaire du patron** : Il faut créer des tests qui sont automatisés, répétables et systématiques. Il faut être capable de se fier aux tests. Ceux-ci démontrent que le système continue de fonctionner normalement. Il est important d'avoir des tests du système actuel avant de faire des modifications. On les appelle aussi des tests de non-régression.

**Résultat Oryx :** Il existe quelques tests pour la persistance dans le code source d'Oryx. Cependant, on remarque qu'ils sont au niveau des « servlets », donc au niveau web. Ce sont plutôt des tests d'intégrations. Il n'existe pas de tests unitaires, surtout au niveau des 6 classes découvertes précédemment. Il sera nécessaire de descendre au niveau de la classe et de certaines méthodes.

#### 8.7.5.3 Ajouter à vos tests par étape (Grow your test base incrementally)

**Sommaire du patron** : il est impossible d'écrire des tests pour tout. Il faut savoir ajouter les tests par étape, selon l'avancement du projet et pour les sections importantes.

**Résultat Oryx** : L'ajout de tests se fera de façon graduelle, en suivant notre niveau de connaissances précises de nos 6 classes.

#### 8.7.5.4 Utiliser un jeu de tests (Use a testing framework)

**Sommaire du patron** : il est préférable de fournir un jeu de tests ou une infrastructure de test pour faciliter et encourager la création de tests

**Résultat Oryx** : La librairie JUNIT est incluse dans le projet. Elle est utilisée dans plusieurs tests, surtout au niveau de l'interface usager. Les tests pour les 6 classes utiliseront ANT.

#### 8.7.5.5 Vérifier le comportement et non son implémentation (Test the Interface, Not the Implementation)

**Sommaire du patron** : Ce patron est connu sous le nom de « black box testing » en anglais. Le principe est d'exécuter des tests sur le comportement externe de l'application. Ce genre de tests survivra les différentes modifications internes au système.

**Résultat Oryx** : Les tests pour Oryx seraient au niveau de la persistance. Spécifiquement, ils seront au niveau des opérations de base qui sont la lecture et l'écriture de modèle dans la base de données.

#### 8.7.5.6 Écrire les tests selon les règles d'affaires (Record Business Rules as Tests)

**Sommaire du patron** : Ce patron encourage l'écriture de test selon les règles d'affaires que nous trouvons lors de nos recherches. Il n'y a aucune garantie que la documentation ni les commentaires dans le code source ne seront précis.

**Résultat Oryx** : Nos 6 classes de persistance ne contiennent pas nécessairement des règles d'affaires, mais elles contiennent la logique sur la façon de stocker les données. Ultérieurement, si l'objectif de la rétro-ingénierie est d'enlever complètement Hibernate, ce patron deviendra encore plus important.

#### 8.7.5.7 Écrire des tests pour comprendre le comportement de l'application (Write Tests to Understand)

**Sommaire du patron** : Ce patron encourage l'écriture de test pour enregistrer notre apprentissage sur le fonctionnement du logiciel.

**Résultat Oryx** : Ultérieurement, si l'objectif de la rétro-ingénierie dépasse la modification de la couche de persistance, ce patron deviendra encore plus important.

#### 8.7.5.8 Stratégie de migration (Migration Strategies)

On retrouve des patrons qui demande l'implication des usagers, créer des prototypes incrémentaux et autres. Nous pouvons reconnaitre l'utilisation de la méthode du cycle itératif définie dans le développement agile.

Cette section sur le groupement de patrons de migration n'est pas nécessaire pour notre projet. Il y aura un petit sommaire pour chaque patron, mais aucune information de son utilisation pour notre projet.

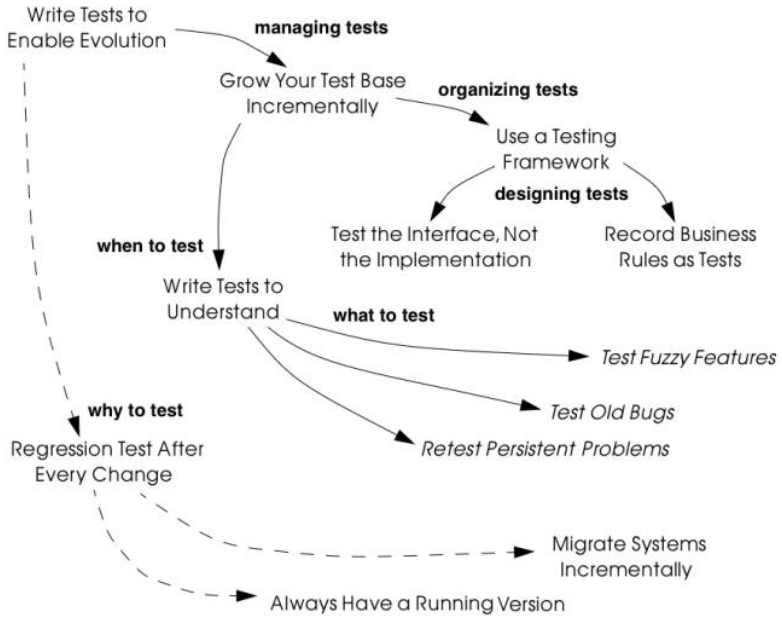


Figure 6.1: When, why, how and what to test.

Figure 60 - regroupement de patrons - la migration (Demeyer, Ducasse, & Nierstrasz, 2002)

#### 8.7.5.9 Impliquer les usagers (involve the users)

**Sommaire du patron** : Il faut impliquer les usagers à chaque étape de ce processus. Ceci maximise les chances d'acceptations des changements.

#### 8.7.5.10 Bâtir la confiance dans le processus (Build Confidence)

**Sommaire du patron** : Il faut utiliser le cycle itératif pour démontrer les changements de façon régulière.

#### 8.7.5.11 Migration par étape (Migrate Systems Incrementally)

**Sommaire du patron** : Il faut minimiser les risques de migration. Il est préférable de faire des petits changements. L'approche « big bang » est déconseillée.

#### 8.7.5.12 Créez un prototype (Prototype the Target Solution)

**Sommaire du patron** : Si c'est applicable, il est possible de créer un prototype pour démontrer les modifications possibles.

#### 8.7.5.13 Faire la construction du logiciel de façon régulière (Always have a running version)

**Sommaire du patron** : Ce patron aide à bâtir la confiance des usagers en démontrant la stabilité du processus de construction du logiciel.

#### 8.7.5.14 Les tests de régression (Regression test after every change)

**Sommaire du patron** : Il faut démontrer que les tests de régression fonctionnent bien. Ces tests démontrent que le logiciel continue de fonctionner normalement.

#### 8.7.5.15 Utilisation des 2 logiciels en parallèle avec un pont (Make a bridge to the new town)

**Sommaire du patron** : Si possible, faire exécuter la nouvelle version de l'application en même temps que l'ancienne, avec possiblement un pont entre les deux pour faire l'échange de données.

#### 8.7.5.16 Écrire une couche isolante autour de l'ancienne application (Present the right interface)

**Sommaire du patron** : Si possible, écrire une couche autour de l'ancienne application pour que la nouvelle soit capable d'extraire les données nécessaires.

#### 8.7.5.17 Établir deux niveaux d'interface (Distinguish public from published interface)

**Sommaire du patron** : Si possible, présenter des interfaces non stables comme étant « disponibles », et les interfaces stables comme « publiques »

#### 8.7.5.18 Annoncez les interfaces désuètes (Deprecate obsolete interfaces)

**Sommaire du patron** : Déclarez les interfaces désuètes comme telles. @deprecated

#### 8.7.5.19 Évitez les changements trop brusques (Conserve familiarity)

**Sommaire du patron** : Évitez les grosses modifications.

#### 8.7.5.20 Identifiez les goulots d'étranglement (Use profiler before optimizing)

**Sommaire du patron** : Choisissez vos efforts d'optimisation.

### 8.7.5.21 La duplication de code (Detecting duplicated code)

La duplication de code est un problème de premier ordre. C'est un facteur qui indique le besoin de restructurer une section de code. La duplication de code ajoute à votre dette technique. Ceci veut dire que c'est un problème que vous devrez corriger dans le futur. Et comme une dette, le plus longtemps que vous attendez, le plus cher que ça va couter.

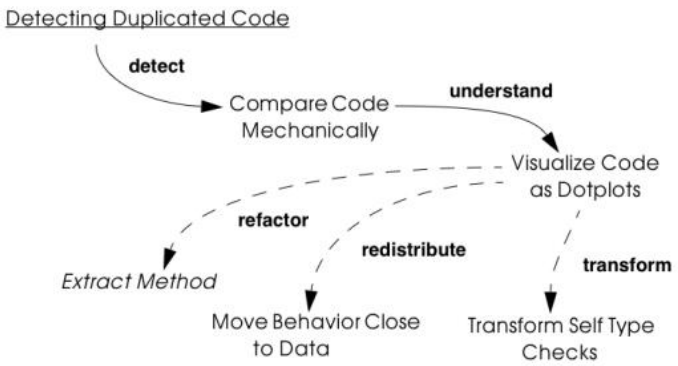


Figure 8.1: Two patterns to support Detecting Duplicated Code.

Figure 61 - regroupement de patrons - détections de duplicata de code

### 8.7.5.22 Comparez le code source (Compare code mechanically)

**Sommaire du patron** : Il existe plusieurs techniques pour trouver la duplication de code.

### 8.7.5.23 Visionner le code avec un graphique (Visualize code as dotplots)

**Sommaire du patron** : Utilisez des outils qui vous permettront de visionner votre code en mode graphique. Ceci vous permettra de voir visuellement des phénomènes ou une organisation que vous pouvez observer de façon répétitive.

#### 8.7.5.24 La redistribution des responsabilités (redistribute responsibilities)

Ce groupement de patrons est utilisé pour corriger le problème de se retrouver avec des responsabilités mal distribué dans le code. Les deux plus gros problèmes qui sont retrouvés sont :

- Le conteneur de data : beaucoup de données, mais très peu de responsabilités (méthode, logique de couche métier, etc.)
- Une classe à tout faire (anglais : god class)

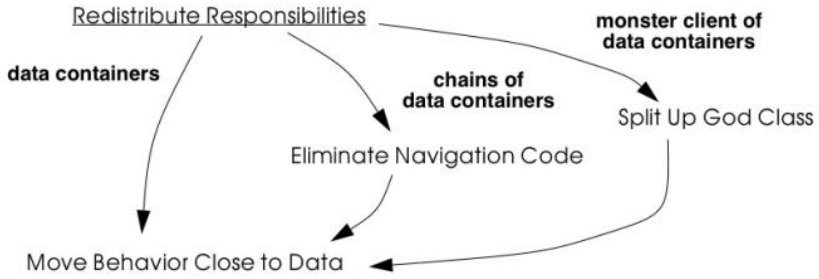


Figure 9.2: Data containers are the clearest sign of misplaced responsibilities. These three patterns redistribute responsibilities by moving behavior close to data.

Figure 62 - regroupement de patrons - redistribution des responsabilités (Demeyer, Ducasse, & Nierstrasz, 2002)

#### 8.7.5.25 Rapatriez la logique plus proche des données (Move behavior close to data)

**Sommaire du patron** : Migrer le code source qui agit sur les données dans les classes qui contiennent ces données.

#### 8.7.5.26 Élimination du code qui ne fait que de la direction (Eliminate Navigation code)

**Sommaire du patron** : Il y a du code source qui ne fait que diriger, sans pour autant accomplir un travail de logique d'affaire, ni de traitement sur les données. Il faut essayer de pousser ce genre de travail le plus proche des objets qui traitent sur les données.

#### 8.7.5.27 La classe à tout faire (Split up God class)

**Sommaire du patron** : Une classe à tout faire a trop de responsabilités, est trop longue et est très difficile à modifier. Un exercice de division des responsabilités en petits morceaux est nécessaire.

#### 8.7.5.28 Réduisez l'utilisation des classes avec de grosses séquences de décision (transform conditionals to polymorphism)

Il faut réduire l'utilisation de classes qui ne font que du cheminement ou décision. Les abus de l'utilisation d'énoncé « IF » ou « CASE » peuvent être réduits avec l'utilisation de polymorphisme.

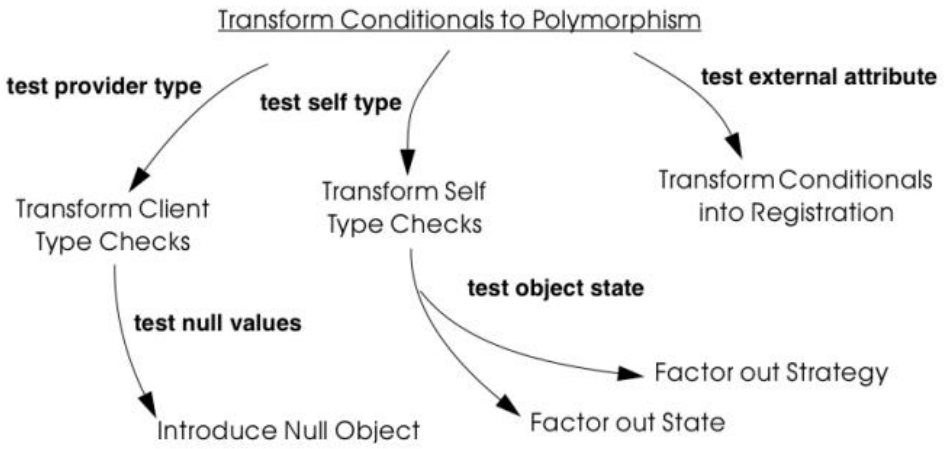


Figure 10.1: Relationships between the patterns constituting Transform Conditionals to Polymorphism.

Figure 63- regroupement de patrons - Conversion vers le polymorphisme (Demeyer, Ducasse, & Nierstrasz, 2002)

#### 8.7.5.29 Remplacement du code conditionnel par un système de “hooks” (Transform self type checks)

**Sommaire du patron** : Lorsqu’il y a une classe avec beaucoup de code conditionnel, il est suggéré d’enlever ce code et le remplacer sur des appels d’une nouvelle méthode dans les sous-classes. Cette nouvelle méthode vérifie si la condition s’applique à elle et réagit en conséquence.

#### 8.7.5.30 Réduisez la vérification du genre de classes (Transform client type checks)

**Sommaire du patron** : Il faut modifier la structure du code si le code semble toujours vérifier quel genre de classe ou données qui existe dans une autre classe.

#### 8.7.5.31 Élimination de la vérification de l'état (Factor out state)

**Sommaire du patron** : Utilisez le patron de conception « state » pour éliminer du code décisionnel trop complexe

#### 8.7.5.32 Élimination du choix d'algorithme (Factor out strategy)

**Sommaire du patron** : Utilisez le patron de conception « Strategy » pour éliminer du code qui choisit un algorithme.

#### 8.7.5.33 Élimination du code qui vérifie pour la valeur "null" (Introduce null object)

**Sommaire du patron** : Utilisez le patron de conception « Null Object » pour éliminer du code qui vérifie pour la valeur « null ».

#### 8.7.5.34 Utilisez un système d'enregistrement (Transform conditionals into registration)

**Sommaire du patron** : Pour améliorer la modularité d'une application, utilisez un système d'enregistrement.

# CHAPITRE 9

## Hibernate – et l'impact sur mon projet

### 9.1 Introduction

La découverte de l'utilisation de Hibernate dans le logiciel Oryx a eu un immense impact sur mon projet. Je m'attendais à travailler avec 'JDBC'. La modification nécessaire pour rediriger Hibernate de PostgreSQL vers une base de données NoSQL a été simple. Il faut faire une petite modification du fichier de configuration XML de Hibernate et un changement de librairies. Pour cette raison, je vais discuter rapidement de cette technologie et clarifier l'impact sur mon projet.

### 9.2 JDBC - définition

JDBC est un acronyme pour « Java Database Connectivity ». C'est une interface de programmation, aussi connue comme API en anglais. On peut retrouver plus d'information sur JDBC sur le site web d'Oracle (Oracle, 2015). Les programmeurs utilisent cette interface pour permettre la persistance des objets, typiquement stockée dans un système de gestion de base de données.

Lors de ce processus de stockage, les objets Java dans l'application doivent être préparés et copiés dans une ou plusieurs structures nommées « prepared statements » pour assurer la transformation de l'objet Java en données relationnelles. Ensuite, les données, maintenant en format relationnel, peuvent être stockées dans une base de données relationnelle.

Le même processus est nécessaire à l'inverse lorsqu'il y a lecture des données, c'est-à-dire, qu'il faut accomplir une transformation des données relationnelle vers des objets Java.
De plus, le code Java doit s'assurer de la cohérence des différentes sortes de données qui sont typiquement stockées dans des tables différentes à l'intérieur d'une base de données

relationnelle. On parle ici de l'utilisation de l'association entre plusieurs tables en utilisant les fonctionnalités SQL, tel que le « JOIN ». Ceci évite la duplication de données, surtout lorsqu'une donnée peut avoir une relation multiple avec une autre sorte de donnée.

## 9.3 Hibernate - définition

Hibernate est une interface de programmation de source libre (Hibernate ORM, 2015) qui ajoute un niveau d'abstraction plus élevé que JDBC. Son objectif est de permettre au programmeur Java de stocker les données en format objet (sans transformation vers le relationnel) dans un système de gestion de base de données relationnelle. Le terme anglais utilisé est « object-relational mapping » ou ORM. En français, la traduction est « mapping objet-relationnel ».

Une interface de programmation « mapping objet-relationel » permet d'associer une classe Java vers une table dans la base de données. L'interface, telle que Hibernate, s'occupe de la transformation « objet-relationnel » en plus d'offrir des opportunités d'amélioration de performance avec l'utilisation d'une architecture de mémoire cache ou antémémoire.

Hibernate offre une configuration simple avec des fichiers XML. Il est utilisable avec de multiples systèmes de gestion de base de données, tel que :

- Oracle,
- DB2,
- MS SQL,
- MySQL,
- PostgreSQL.

Hibernate offre aussi une variante pour les bases de données NoSQL, nommé Hibernate OGM (Hibernate OGM, 2015). Selon la description du site web, Hibernate OGM fait le mapping entre les classes Java et les bases de données NoSQL.

## 9.4 Hibernate – avantages et défis

Comme toutes technologies, il faut savoir l'utiliser dans un contexte qui est propice à son bon usage et à son succès. Hibernate n'échappe pas à cette règle et apporte beaucoup d'avantages à un projet lorsqu'il est bien utilisé.

Plusieurs auteurs nous offrent les commentaires suivants (Begoli, 2015) (Emailed, 2010) (Kim, 2008) (Santosh, 2012):

- Une bonne utilisation de Hibernate peut offrir une amélioration de performance assez remarquable. Les librairies Hibernate génèrent des requêtes SQL très efficace et comme mentionnée dans un paragraphe ci-haut, utilise un stratagème avancé de mémoire cache,
- La portabilité : Hibernate fonctionne avec plusieurs systèmes de gestion de bases de données relationnelles. Aucune modification au code source Java n'est nécessaire pour changer de système de base de données. Une simple modification d'une ligne est nécessaire pour accomplir ceci,
- L'apprentissage de la technologie est difficile, mais court,
- Une fois qu'un programmeur maitrise cette technologie, sa productivité augmente de façon significative.

On peut trouver aussi des situations ou l'utilisation de Hibernate peut être un défi.

- Le créateur de Hibernate, Gavin King (Java Performance Tuning, 2015), mentionne qu'il ne croit pas que Hibernate devrait être utilisé avec des bases de données avec moins de 10 tables. Il mentionne que Hibernate démontre sa puissance et sa capacité lorsqu'on l'utilise avec des modèles de données complexes, des centaines de tables et des relations complexes. La structure de la base de données de l'application Oryx ne

justifie certainement pas l'utilisation de Hibernate, mais je dois conclure que c'était un défi éducationnel,

- Si la quantité des données est très large (million et même milliard), la performance et la fonctionnalité de Hibernate sont beaucoup réduites. L'utilisation de JDBC est suggérée (Santosh, 2012)
- J'ai déjà mentionné que l'utilisation de Big Data est prescrite quand le volume de donnée est très grand. J'ai trouvé intéressant un article (Wohlgemuth, 2010) qui décrit les efforts pour traiter plus de 400 milliards de records dans une base de données PostgreSQL avec Hibernate. Tout va bien jusqu'au seuil de 1 milliard de records dans une table. La performance est inacceptable après ceci. Pour dépasser ce nombre de records, il décrit toutes les étapes et les tests qu'il a faits pour accomplir ceci avec deux résultats à noter :
    - Il a été obligé de séparer sa table en multiples morceaux, aussi connue par le nom anglais de « sharding ou partitionning »,
    - Il a enlevé complètement Hibernate dans son application, car il ne pouvait plus utiliser les atouts de Hibernate après tous les changements qu'il a faits dans son application.
- De l'autre côté de la médaille, un autre auteur (Kuzmin, 2010) décrit les raisons pour ne pas utiliser Hibernate. Un de ses commentaires attire mon attention immédiatement : « Hibernates does not allow refactoring ». Ayant déjà participé à un projet qui utilisait Hibernate, je suis entièrement d'accord avec ce commentaire. Il a été plus rapide pour nous de complètement enlever Hibernate de notre projet.

## 9.5 En prévision de la conversion JDBC avec Oryx

Comme discuté dans la section 1 - la revue de la littérature, et la section 2 – apprentissage de la technologie NoSQL, les bases de données du genre NoSQL n'utilisent pas de SQL, ou de données relationnelles.

Le défi de migrer Oryx de PostgreSQL/JDBC version NoSQL aurait été intéressant, car une recherche aurait été nécessaire pour décider sur la stratégie de stockage avec NoSQL. Les questions suivantes auraient été des sujets de recherche :

- Quel genre de base de données NoSQL devrait être utilisé?
    - Possiblement du genre orienté document pour y stockée un document (carte du processus d'affaires créé par Oryx) en format JSON??
    - Ou possiblement l'utilisation d'une base de données orientée graffe pour insérer chaque item sur la carte du processus d'affaires créer par Oryx dans une chaine (un item dans le processus d'affaires représenté par un nœud).
- Quel format de stockage utilisé pour un processus d'affaires créé par Oryx?
    - En un document XML ou JSON,
    - Multiples données, une pour chaque nœud.

Cependant, l'utilisation de Hibernate élimine complètement ces sujets de recherche, car il suffit d'accomplir un petit changement du fichier de configuration de Hibernate, et le changement de librairie de Hibernate-ORM par Hibernate-OGM.

Je discuterai de ce point dans la « section 4 – interprétation ».

# CONCLUSION

Cette quatrième et dernière section de ce rapport consiste à faire une interprétation de la problématique de recherche et les analyses de la solution proposée. L'objectif est d'en tirer et présenter des conclusions et de faire une évaluation du potentiel pour utilisation futur.

## Discussion concernant les résultats des essais

### NoSQL

L'apprentissage de cette technologie et de son écosystème a été très apprécié. Je comprends maintenant l'ampleur de cette solution, surtout au niveau de l'administration de système et la gestion et l'analyse des données, incluant le filtrage et les efforts requis pour produire des résultats concrets. J'ai acquis une appréciation pour cette technologie et je réalise que mon apprentissage ne fait que commencer. Je note aussi le potentiel des problèmes causés par les néophytes qui ne font pas autant d'apprentissages sur le sujet et offre une solution NoSQL médiocre, simplement pour être capable d'ajouter ce mot sur une offre de service.

### OORP

Il est évident que l'approche OORP démontre son potentiel lorsqu'il y a une équipe avec plusieurs membres. Il y a plusieurs sections qui étaient simplement inutiles pour moi, surtout dans la première section. Cependant, j'ai été agréablement surpris de la façon ordonnée du processus.

Sans l'utilisation de cette approche, j'aurai surement fait la majorité de ces étapes, mais la séquence m'a permis de faire cette analyse de façon ordonnée. De plus, je reconnais la capacité de cette approche de négocier les influences, les objectifs cachés et les distractions qui font toujours partie de n'importe quel genre de projet. J'ajoute OORP à mon éventail d'outils dans le futur.

Oryx

La section 3 décrit la découverte de l'utilisation de Hibernate et son impact sur mon projet. C'était inattendu et décevant. Une application utilise la fonctionnalité de Hibernate pour écrire ses données en conservant la structure objet des données au lieu de forcer une conversion vers le relationnel, chose qui est nécessaire lorsque l'on utilise JDBC. Un des effets immédiats est que Hibernate isole l'application de toutes notions du système de base de données. C'est pour cette raison que l'on peut changer le système de base de données sans modification de l'application. Il suffit de faire un changement de librairie et de faire une modification au fichier .XML de la configuration de Hibernate. L'attente était plutôt de modifier les 6 classes identifiées et remplacer les appels JDBC par les appels « get, put, scan, delete » de HBase.

Il y a cependant deux bémols à cette découverte. La première est que Hibernate est compatible avec plusieurs types de bases de données NoSQL, mais l'utilisation du genre « column family », tel que HBase et Cassandra sont encore en mode développement. Le deuxième bémol est que la page des projets disponibles sur le site de l'ÉTS mentionnait qu'il y avait un goulot d'étranglement lors de la lecture et l'écriture des modèles stockée dans la base de données. La correction de ce problème ne faisant pas partie de mes objectifs, je n'ai fait aucune mesure à cet effet. Cependant, j'ai déjà mentionné que l'utilisation de Hibernate pour une base de données avec peu de tables est découragée.

Possiblement un travail pour complètement enlever Hibernate et utiliser des appels natifs (JDBC ou les appels « get, put, scan, delete » des librairies de HBase pourront fournir des améliorations de performance.

En conclusion, j'ai apprécié ce projet et je vais certainement utiliser ces nouvelles connaissances dans mes projets. En plus des commentaires précédents, je retiens particulièrement les informations que j'ai recueillies au sujet de l'étape de la compréhension d'un logiciel et je vais continuer mes efforts pour accroître le transfert du savoir à l'intérieur de mon entreprise. De plus, je vais étudier en plus de détail la notion de la gestion des processus

d’affaires intelligents (iBPM) pour évaluer la possibilité d’ajouter ce genre de fonctionnalité dans nos produits.

# ANNEXE I

## Tutoriel #3 – example #1 de Xu Fei

```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.client.Get;
import org.apache.hadoop.hbase.client.HTable;
import org.apache.hadoop.hbase.client.Put;
import org.apache.hadoop.hbase.client.Result;
import org.apache.hadoop.hbase.client.ResultScanner;
import org.apache.hadoop.hbase.client.Scan;
import org.apache.hadoop.hbase.util.Bytes;

public class HBaseConnector {
  public static void main(String[] args) throws IOException {
    // You need a configuration object to tell the client where to connect.
    // When you create a HBaseConfiguration, it reads in whatever you've set
    // into your hbase-site.xml and in hbase-default.xml, as long as these
    // can be found on the CLASSPATH
    System.out.println("creating configuration object");
    Configuration config = HBaseConfiguration.create();

    // This instantiates an HTable object that connects you to the
    // "myLittleHBaseTable" table.
    System.out.println("creating HTable object");
    HTable table = new HTable(config, "myLittleHBaseTable");

    // To add to a row, use Put. A Put constructor takes the name of the row
    // you want to insert into as a byte array. In HBase, the Bytes class
    // has utility for converting all kinds of java types to byte arrays. In
    // the below, we are converting the String "myLittleRow" into a byte
    // array to use as a row key for our update. Once you have a Put
    // instance, you can adorn it by setting the names of columns you want
    // to update on the row, the timestamp to use in your update, etc.
    // If no timestamp, the server applies current time to the edits.
    System.out.println("creating PUT object");
    Put p = new Put(Bytes.toBytes("myLittleRow"));

    // To set the value you'd like to update in the row 'myLittleRow',
    // specify the column family, column qualifier, and value of the table
    // cell you'd like to update. The column family must already exist
    // in your table schema. The qualifier can be anything.
    // All must be specified as byte arrays as hbase is all about byte
    // arrays. Lets pretend the table 'myLittleHBaseTable' was created
    // with a family 'myLittleFamily'.
    System.out.println("updating PUT object");
    p.add(Bytes.toBytes("myLittleFamily"), Bytes.toBytes("someQualifier"),
        Bytes.toBytes("Some Value"));

    // Once you've adorned your Put instance with all the updates you want
    // to make, to commit it do the following
    // (The HTable#put method takes the Put instance you've been building
    // and pushes the changes you made into hbase)
    System.out.println("committing to table");
    table.put(p);

    // Now, to retrieve the data we just wrote. The values that come back
```

```
    // are Result instances. Generally, a Result is an object that will
    // package up the hbase return into the form you find most palatable.
    System.out.println("creating GET object");
    Get g = new Get(Bytes.toBytes("myLittleRow"));
    System.out.println("get row and put in Result");
    Result r = table.get(g);
    byte[] value = r.getValue(Bytes.toBytes("myLittleFamily"), Bytes
        .toBytes("someQualifier"));
    // If we convert the value bytes, we should get back 'Some Value', the
    // value we inserted at this location.
    String valueStr = Bytes.toString(value);
    System.out.println("GET: " + valueStr);

    // Sometimes, you won't know the row you're looking for. In this case,
    // you use a Scanner. This will give you cursor-like interface to the
    // contents of the table. To set up a Scanner, do like you did above
    // making a Put and a Get, create a Scan. Adorn it with column names,
    // etc.
    Scan s = new Scan();
    s.addColumn(Bytes.toBytes("myLittleFamily"), Bytes
        .toBytes("someQualifier"));
    ResultScanner scanner = table.getScanner(s);
    try {
      // Scanners return Result instances.
      // Now, for the actual iteration. One way is to use a while loop
      // like so:
      for (Result rr = scanner.next(); rr != null; rr = scanner.next()) {
        // print out the row we found and the columns we were looking
        // for
        System.out.println("Found row: " + rr);
      }

      // The other approach is to use a foreach loop. Scanners are
      // iterable!
      // for (Result rr : scanner) {
      // System.out.println("Found row: " + rr);
      // }
    } finally {
      // Make sure you close your scanners when you are done!
      // Thats why we have it inside a try/finally clause
      scanner.close();
    }
    table.close();
  }
}
```

# ANNEXE II

## Tutoriel #3 – exemple #2 de Xu Fei

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.HColumnDescriptor;
import org.apache.hadoop.hbase.HTableDescriptor;
import org.apache.hadoop.hbase.KeyValue;
import org.apache.hadoop.hbase.MasterNotRunningException;
import org.apache.hadoop.hbase.ZooKeeperConnectionException;
import org.apache.hadoop.hbase.client.Delete;
import org.apache.hadoop.hbase.client.Get;
import org.apache.hadoop.hbase.client.HBaseAdmin;
import org.apache.hadoop.hbase.client.HTable;
import org.apache.hadoop.hbase.client.Result;
import org.apache.hadoop.hbase.client.ResultScanner;
import org.apache.hadoop.hbase.client.Scan;
import org.apache.hadoop.hbase.client.Put;
import org.apache.hadoop.hbase.util.Bytes;

public class HBaseTest {

  private static Configuration conf = null;
  /**
   * Initialization
   */
  static {
    conf = HBaseConfiguration.create();
  }

  /**
   * Create a table
   */
  public static void creatTable(String tableName, String[] familys)
      throws Exception {
    System.out.println("new admin");
    HBaseAdmin admin = new HBaseAdmin(conf);
    System.out.println("new admin after");
    if (admin.tableExists(tableName)) {
      System.out.println("table already exists!");
    } else {
      System.out.println("new htabledescriptor");
      HTableDescriptor tableDesc = new HTableDescriptor(tableName);
      for (int i = 0; i < familys.length; i++) {
        tableDesc.addFamily(new HColumnDescriptor(familys[i]));
      }
      admin.createTable(tableDesc);
      System.out.println("create table " + tableName + " ok.");
    }
  }

  /**
   * Delete a table
   */
  public static void deleteTable(String tableName) throws Exception {
    try {
      HBaseAdmin admin = new HBaseAdmin(conf);
      admin.disableTable(tableName);
```

```
    admin.deleteTable(tableName);
    System.out.println("delete table " + tableName + " ok.");
  } catch (MasterNotRunningException e) {
    e.printStackTrace();
  } catch (ZooKeeperConnectionException e) {
    e.printStackTrace();
  }
}

/**
 * Put (or insert) a row
 */
public static void addRecord(String tableName, String rowKey,
    String family, String qualifier, String value) throws Exception {
  try {
    HTable table = new HTable(conf, tableName);
    Put put = new Put(Bytes.toBytes(rowKey));
    put.add(Bytes.toBytes(family), Bytes.toBytes(qualifier), Bytes
        .toBytes(value));
    table.put(put);
    System.out.println("insert recored " + rowKey + " to table "
        + tableName + " ok.");
  } catch (IOException e) {
    e.printStackTrace();
  }
}

/**
 * Delete a row
 */
public static void delRecord(String tableName, String rowKey)
    throws IOException {
  HTable table = new HTable(conf, tableName);
  List<Delete> list = new ArrayList<Delete>();
  Delete del = new Delete(rowKey.getBytes());
  list.add(del);
  table.delete(list);
  System.out.println("del recored " + rowKey + " ok.");
}

/**
 * Get a row
 */
public static void getOneRecord (String tableName, String rowKey) throws IOException{
      HTable table = new HTable(conf, tableName);
      Get get = new Get(rowKey.getBytes());
      Result rs = table.get(get);
      for(KeyValue kv : rs.raw()){
          System.out.print(new String(kv.getRow()) + " " );
          System.out.print(new String(kv.getFamily()) + ":" );
          System.out.print(new String(kv.getQualifier()) + " " );
          System.out.print(kv.getTimestamp() + " " );
          System.out.println(new String(kv.getValue()));
      }
  }
/**
 * Scan (or list) a table
 */
public static void getAllRecord (String tableName) {
      try{
           HTable table = new HTable(conf, tableName);
           Scan s = new Scan();
           ResultScanner ss = table.getScanner(s);
           for(Result r:ss){
               for(KeyValue kv : r.raw()){
                  System.out.print(new String(kv.getRow()) + " ");
```

```
                    System.out.print(new String(kv.getFamily()) + ":");
                    System.out.print(new String(kv.getQualifier()) + " ");
                    System.out.print(kv.getTimestamp() + " ");
                    System.out.println(new String(kv.getValue()));
                }
            }
        } catch (IOException e){
            e.printStackTrace();
        }
    }

  public static void main(String[] agrs) {
    try {
      String tablename = "scores";
      String[] familys = { "grade", "course" };
      System.out.println("create a table");
      HBaseTest.creatTable(tablename, familys);

      System.out.println("add records");
      // add record zkb
      HBaseTest.addRecord(tablename, "zkb", "grade", "", "5");
      HBaseTest.addRecord(tablename, "zkb", "course", "", "90");
      HBaseTest.addRecord(tablename, "zkb", "course", "math", "97");
      HBaseTest.addRecord(tablename, "zkb", "course", "art", "87");
      // add record baoniu
      HBaseTest.addRecord(tablename, "baoniu", "grade", "", "4");
      HBaseTest.addRecord(tablename, "baoniu", "course", "math", "89");

      System.out.println("===========get one record========");
      HBaseTest.getOneRecord(tablename, "zkb");

      System.out.println("===========show all record========");
      HBaseTest.getAllRecord(tablename);

      System.out.println("===========del one record========");
      HBaseTest.delRecord(tablename, "baoniu");
      HBaseTest.getAllRecord(tablename);

      System.out.println("===========show all record========");
      HBaseTest.getAllRecord(tablename);
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

# ANNEXE III

## Cadre de Basili

### 9.6 Définition

La phase de définition consiste en l'identification de la problématique de recherche et des pistes de solutions sont explorer pour un estimé d'effort de la recherche.

| Élement | Description |
|---|---|
| **Motivation** | • Approfondir connaissance dans la rétro-ingénierie du logiciel<br>• Apprendre une nouvelle technologie : Big Data<br>• Expérimentation dans le domaine de la gestion de processus d'affaires (BPM) |
| **But** | • Effectuer la migration du SGBD PostgreSQL utilisé dans le logiciel Oryx par une BD No-SQL (HBASE) utilisée dans les nouvelles technologies BigData<br>• Expérimentation d'un processus de rétro-ingénierie |
| **Objet** | • Orxy (Oryx-Editor, s.d.), un système web pour la création et modification de schéma de processus d'affaire<br>• Système de gestion de base de données : PostgreSQL et BigData tel que Hadoop<br>• Processus de rétro-ingénierie |
| **Perspective** | • De la perspective d'un gestionnaire et architecte applicatif |
| **Domaine** | • Tel qu'elles seront appliqués par un ingénieur logiciel d'expérience |
| **Étendue** | • Expérience en laboratoire sur un logiciel libre |

## 9.7 Planification

La phase de planification identifie les livrables nécessaire à la résolution de la problématique et répondre aux questions de recherche.

| Étape | Activités | Résultats/Mesure |
|---|---|---|
| Revue de la littérature<br>• Système de gestion des processus d'affaires<br>• rétro-ingénierie de logiciel<br>• BigData | Choix d'article et lecture de ceux-ci | Rapport sur la littérature (première partie du rapport) |
| Apprentissage de la technologie BigData | • Installation, configuration d'une BD BigData et développement d'un (ou plusieurs) application(s) démonstrateur(s)<br>• Suivre tutoriel disponible | Installation fonctionnelle de HBASE avec logiciel(s) qui démontre(nt) son utilisation (deuxième partie du rapport) |
| Rétro-ingénierie de l'application Oryx en utilisant l'approche « Object-Oriented Reengineering Patterns » OORP (Demeyer, Ducasse, & Nierstrasz, 2002) | En suivant la méthodologie (ou patron de conception) OORP, faire fonctionner le logiciel Oryx avec une BD no-sql (Hbase) de type BigData | Application de technique de rétro-ingénierie dans le contexte de technologies BigData (troisième partie du rapport) |

### 9.8 Exécution/Développement

La phase de développement établie les composantes qui permettrons de formé la solution à la question principale de recherche.

| Préparation | Éxécution | Analyse |
|---|---|---|
| Obtention d'un environnement de serveur pour système BigData | -créer l'environnement BigData<br>-modification/rétro-ingénierie de l'application Oryx | -journal des activités de l'approche OORP<br>-ébauche d'un document d'architecture selon ISO 42010<br>-ébauche d'un document « notes de mise à jour » |
| Développement de cas d'essais pour l'expérimentation | Exécution des essais | Documentation des résultats des essais et problématiques |

### 9.9 Interprétation

La phase d'interprétation consiste finalement à revoir la problématique de recherche et les analyses de la solution proposé pour en tirer des conclusions, évalué l'intérêt de la solution proposé pour l'industrie et finalement d'identifiées les future travaux. Ceci est la quatrième et dernière section du rapport de projet.

| Contexte | Extrapolation | Travaux futures |
|---|---|---|
| Discussion concernant les résultats des essais | Discussion de l'utilisation de cette approche à grande échelle | -comment utiliser ces nouvelles connaissances dans le futur<br>-comment améliorer le processus utilisé |

# ANNEXE IV

## Oryx – schéma de base de données

```
--
-- PostgreSQL database dump
--
SET client_encoding = 'UTF8';
SET standard_conforming_strings = off;
SET check_function_bodies = false;
SET client_min_messages = warning;
SET escape_string_warning = off;
--
-- Name: plpgsql; Type: PROCEDURAL LANGUAGE; Schema: -; Owner: poem
--
CREATE PROCEDURAL LANGUAGE plpgsql;

ALTER PROCEDURAL LANGUAGE plpgsql OWNER TO poem;
--
-- Name: plpythonu; Type: PROCEDURAL LANGUAGE; Schema: -; Owner: poem
--
CREATE PROCEDURAL LANGUAGE plpythonu;

ALTER PROCEDURAL LANGUAGE plpythonu OWNER TO poem;

SET search_path = public, pg_catalog;
--
-- Name: is_parent(text, text); Type: FUNCTION; Schema: public; Owner: poem
--
CREATE FUNCTION is_parent(node text, subnode text) RETURNS boolean
    AS $$BEGIN
 PERFORM * FROM poem_path(subnode) WHERE poem_path=node;
 IF NOT FOUND THEN
        RETURN FALSE;
 ELSE
        RETURN TRUE;
 END IF;
END;$$
    LANGUAGE plpgsql IMMUTABLE;

ALTER FUNCTION public.is_parent(node text, subnode text) OWNER TO poem;

--
-- Name: parent(text); Type: FUNCTION; Schema: public; Owner: poem
--

CREATE FUNCTION parent(hierarchy text) RETURNS text
    AS $$
 def poem_path(hierarchy):
        # Returns an ordered collection of strings. The first item of the collection is
        # the path to the first item in the hierarchy and so on.
        all = ["",""]
        if (hierarchy == ""):
                return all
        else:
                position = 0
                # for each character in the input string
                while position < len(hierarchy):
                        # count tildes
                        tilde_count = 0
                        while hierarchy[position+tilde_count] == "~":
                                tilde_count += 1
```

```
                        if tilde_count == 0:
                                all.append(hierarchy[:position+1])
                                position += 1
                        else:
                                all.append(hierarchy[:position+2*tilde_count])
                        position += 2*tilde_count;
                return all

 # Returns path to parent item
 return poem_path(hierarchy)[-2]
 $$
    LANGUAGE plpythonu IMMUTABLE;


ALTER FUNCTION public.parent(hierarchy text) OWNER TO poem;

SET default_tablespace = '';

SET default_with_oids = false;

--
-- Name: identity; Type: TABLE; Schema: public; Owner: poem; Tablespace:
--

CREATE TABLE identity (
    id integer NOT NULL,
    uri text NOT NULL
);
ALTER TABLE ONLY identity ALTER COLUMN uri SET STORAGE MAIN;


ALTER TABLE public.identity OWNER TO poem;

--
-- Name: interaction; Type: TABLE; Schema: public; Owner: poem; Tablespace:
--

CREATE TABLE interaction (
    id integer NOT NULL,
    subject text NOT NULL,
    subject_descend boolean DEFAULT false NOT NULL,
    object text NOT NULL,
    object_self boolean DEFAULT true NOT NULL,
    object_descend boolean DEFAULT false NOT NULL,
    object_restrict_to_parent boolean DEFAULT false NOT NULL,
    scheme text NOT NULL,
    term text NOT NULL
);
ALTER TABLE ONLY interaction ALTER COLUMN subject SET STORAGE MAIN;
ALTER TABLE ONLY interaction ALTER COLUMN object SET STORAGE MAIN;
ALTER TABLE ONLY interaction ALTER COLUMN scheme SET STORAGE MAIN;
ALTER TABLE ONLY interaction ALTER COLUMN term SET STORAGE MAIN;


ALTER TABLE public.interaction OWNER TO poem;

--
-- Name: structure; Type: TABLE; Schema: public; Owner: poem; Tablespace:
--

CREATE TABLE structure (
    hierarchy text NOT NULL,
    ident_id integer NOT NULL
);
ALTER TABLE ONLY structure ALTER COLUMN hierarchy SET STORAGE MAIN;
```

```
ALTER TABLE public.structure OWNER TO poem;

--
-- Name: access; Type: VIEW; Schema: public; Owner: postgres
--

CREATE OR REPLACE VIEW access AS
 SELECT context_name.id AS context_id, context_name.uri AS context_name, subject_name.id AS
subject_id, subject_name.uri AS subject_name, object_name.id AS object_id, object_name.uri AS
object_name, access.id AS access_id, access.scheme AS access_scheme, access.term AS access_term
   FROM interaction access, structure context, identity context_name, structure subject_axis,
identity subject_name, structure object_axis, identity object_name
  WHERE access.subject = context.hierarchy AND context.ident_id = context_name.id AND
(access.subject = subject_axis.hierarchy OR access.subject_descend AND is_parent(
access.subject, subject_axis.hierarchy)) AND (NOT access.object_restrict_to_parent AND
access.object_self AND access.object = object_axis.hierarchy OR NOT
access.object_restrict_to_parent AND access.object_descend AND is_parent(access.object,
object_axis.hierarchy) OR access.object_restrict_to_parent AND access.object_self AND
object_axis.hierarchy = subject_axis.hierarchy OR access.object_restrict_to_parent AND
access.object_descend AND parent(object_axis.hierarchy) = subject_axis.hierarchy) AND
subject_axis.ident_id = subject_name.id AND object_axis.ident_id = object_name.id;


ALTER TABLE public.access OWNER TO poem;

--
-- Name: comment_id_seq; Type: SEQUENCE; Schema: public; Owner: poem
--

CREATE SEQUENCE comment_id_seq
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;


ALTER TABLE public.comment_id_seq OWNER TO poem;

--
-- Name: comment_id_seq; Type: SEQUENCE SET; Schema: public; Owner: poem
--

SELECT pg_catalog.setval('comment_id_seq', 1, false);


--
-- Name: comment; Type: TABLE; Schema: public; Owner: poem; Tablespace:
--

CREATE TABLE comment (
    id integer DEFAULT nextval('comment_id_seq'::regclass) NOT NULL,
    subject_id integer NOT NULL,
    title text,
    content text NOT NULL
);


ALTER TABLE public.comment OWNER TO poem;

--
-- Name: content; Type: TABLE; Schema: public; Owner: poem; Tablespace:
--

CREATE TABLE content (
```

```
    id integer NOT NULL,
    erdf text NOT NULL,
    svg text,
    png_large bytea,
    png_small bytea
);
ALTER TABLE ONLY content ALTER COLUMN erdf SET STORAGE MAIN;
ALTER TABLE ONLY content ALTER COLUMN svg SET STORAGE MAIN;


ALTER TABLE public.content OWNER TO poem;

--
-- Name: friend; Type: TABLE; Schema: public; Owner: poem; Tablespace:
--

CREATE TABLE friend (
    id integer NOT NULL,
    subject_id integer NOT NULL,
    friend_id integer NOT NULL,
    model_count integer DEFAULT 0 NOT NULL
);


ALTER TABLE public.friend OWNER TO poem;

--
-- Name: model_rating; Type: TABLE; Schema: public; Owner: poem; Tablespace:
--

CREATE TABLE model_rating (
    id integer NOT NULL,
    subject_id integer NOT NULL,
    object_id integer NOT NULL,
    score integer NOT NULL
);


ALTER TABLE public.model_rating OWNER TO poem;

--
-- Name: plugin; Type: TABLE; Schema: public; Owner: poem; Tablespace:
--

CREATE TABLE plugin (
    rel text NOT NULL,
    title text NOT NULL,
    description text NOT NULL,
    java_class text NOT NULL,
    is_export boolean NOT NULL
);
ALTER TABLE ONLY plugin ALTER COLUMN rel SET STORAGE MAIN;
ALTER TABLE ONLY plugin ALTER COLUMN title SET STORAGE MAIN;
ALTER TABLE ONLY plugin ALTER COLUMN description SET STORAGE MAIN;
ALTER TABLE ONLY plugin ALTER COLUMN java_class SET STORAGE MAIN;


ALTER TABLE public.plugin OWNER TO poem;

--
-- Name: representation; Type: TABLE; Schema: public; Owner: poem; Tablespace:
--

CREATE TABLE representation (
    id integer NOT NULL,
    ident_id integer NOT NULL,
```

```
    mime_type text NOT NULL,
    language text DEFAULT 'en_US'::text NOT NULL,
    title text DEFAULT ''::text NOT NULL,
    summary text DEFAULT ''::text NOT NULL,
    created timestamp with time zone DEFAULT now() NOT NULL,
    updated timestamp with time zone DEFAULT now() NOT NULL,
    type text DEFAULT 'undefined'::text NOT NULL
);
ALTER TABLE ONLY representation ALTER COLUMN mime_type SET STORAGE MAIN;
ALTER TABLE ONLY representation ALTER COLUMN language SET STORAGE MAIN;
ALTER TABLE ONLY representation ALTER COLUMN title SET STORAGE MAIN;
ALTER TABLE ONLY representation ALTER COLUMN summary SET STORAGE MAIN;


ALTER TABLE public.representation OWNER TO poem;

--
-- Name: schema_info; Type: TABLE; Schema: public; Owner: poem; Tablespace:
--

CREATE TABLE schema_info (
    version integer
);


ALTER TABLE public.schema_info OWNER TO poem;

--
-- Name: setting_id_seq; Type: SEQUENCE; Schema: public; Owner: poem
--

CREATE SEQUENCE setting_id_seq
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;


ALTER TABLE public.setting_id_seq OWNER TO poem;

--
-- Name: setting_id_seq; Type: SEQUENCE SET; Schema: public; Owner: poem
--

SELECT pg_catalog.setval('setting_id_seq', 3, true);


--
-- Name: setting; Type: TABLE; Schema: public; Owner: poem; Tablespace:
--

CREATE TABLE setting (
    subject_id integer,
    id integer DEFAULT nextval('setting_id_seq'::regclass) NOT NULL,
    key text NOT NULL,
    value text NOT NULL
);


ALTER TABLE public.setting OWNER TO poem;

--
-- Name: subject; Type: TABLE; Schema: public; Owner: poem; Tablespace:
--

CREATE TABLE subject (
```

```
    ident_id integer NOT NULL,
    nickname text,
    email text,
    fullname text,
    dob date,
    gender text,
    postcode text,
    first_login date NOT NULL,
    last_login date NOT NULL,
    login_count integer DEFAULT 0 NOT NULL,
    language_code text,
    country_code text,
    password text,
    visibility text
);


ALTER TABLE public.subject OWNER TO poem;

--
-- Name: tag_definition_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
--

CREATE SEQUENCE tag_definition_id_seq
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;


ALTER TABLE public.tag_definition_id_seq OWNER TO poem;

--
-- Name: tag_definition_id_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres
--

SELECT pg_catalog.setval('tag_definition_id_seq', 1, true);


--
-- Name: tag_definition; Type: TABLE; Schema: public; Owner: poem; Tablespace:
--

CREATE TABLE tag_definition (
    id integer DEFAULT nextval('tag_definition_id_seq'::regclass) NOT NULL,
    subject_id integer NOT NULL,
    name text NOT NULL
);


ALTER TABLE public.tag_definition OWNER TO poem;

--
-- Name: tag_relation_id_seq; Type: SEQUENCE; Schema: public; Owner: poem
--

CREATE SEQUENCE tag_relation_id_seq
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;


ALTER TABLE public.tag_relation_id_seq OWNER TO poem;

--
```

```
-- Name: tag_relation_id_seq; Type: SEQUENCE SET; Schema: public; Owner: poem
--

SELECT pg_catalog.setval('tag_relation_id_seq', 1, true);


--
-- Name: tag_relation; Type: TABLE; Schema: public; Owner: poem; Tablespace:
--

CREATE TABLE tag_relation (
    id integer DEFAULT nextval('tag_relation_id_seq'::regclass) NOT NULL,
    tag_id integer NOT NULL,
    object_id integer NOT NULL
);


ALTER TABLE public.tag_relation OWNER TO poem;

--
-- Name: child_position(text); Type: FUNCTION; Schema: public; Owner: poem
--

CREATE FUNCTION child_position(hierarchy text) RETURNS integer
    AS $$
 def poem_path(hierarchy):
        # Returns an ordered collection of strings. The first item of the collection is
        # the path to the first item in the hierarchy and so on.
        all = ["",""]
        if (hierarchy == ""):
                return all
        else:
                position = 0
                # for each character in the input string
                while position < len(hierarchy):
                        # count tildes
                        tilde_count = 0
                        while hierarchy[position+tilde_count] == "~":
                                tilde_count += 1
                        if tilde_count == 0:
                                all.append(hierarchy[:position+1])
                                position += 1
                        else:
                                all.append(hierarchy[:position+2*tilde_count])
                        position += 2*tilde_count;
                return all

 def decode_position(code):
        # Decodes the input code string to the original number
        offset = 0
        if (code == ""):
                return None
        while (code[offset] == "~"):
            offset+=1
        digits = []
        for digit in code[offset:]:
                digit_no = ord(digit)
                if digit_no in range(48,58):
                        digits.append(digit_no-48)
                if digit_no in range(65, 91):
                        digits.append(digit_no-55)
                if digit_no in range(97,123):
                        digits.append(digit_no-61)

        if offset ==  0:
            return digits[0]
```

```
       if offset ==  1:
           return digits[0] + 62
       if offset ==  2:
           return 124 + digits[0] * 62 + digits[1]
       if offset == 3:
              return 3968 + digits[0] * 62 * 62 + digits[1] * 62 + digits[2]

 path = poem_path(hierarchy)
 parent = -2
 current = -1
 local = path[current][len(path[parent]):] # == current-parent
 return decode_position(local)
 $$
    LANGUAGE plpythonu IMMUTABLE;


ALTER FUNCTION public.child_position(hierarchy text) OWNER TO poem;

--
-- Name: decode_position(text); Type: FUNCTION; Schema: public; Owner: poem
--

CREATE FUNCTION decode_position(code text) RETURNS integer
    AS $$
 # Decodes the input code string to the original number
 offset = 0
 if (code == ""):
       return None
 while (code[offset] == "~"):
     offset+=1
 digits = []
 for digit in code[offset:]:
       digit_no = ord(digit)
       if digit_no in range(48,58):
              digits.append(digit_no-48)
       if digit_no in range(65, 91):
              digits.append(digit_no-55)
       if digit_no in range(97,123):
              digits.append(digit_no-61)

 if offset ==  0:
     return digits[0]
 if offset ==  1:
     return digits[0] + 62
 if offset ==  2:
     return 124 + digits[0] * 62 + digits[1]
 if offset == 3:
       return 3968 + digits[0] * 62 * 62 + digits[1] * 62 + digits[2]
 $$
    LANGUAGE plpythonu IMMUTABLE;


ALTER FUNCTION public.decode_position(code text) OWNER TO poem;

--
-- Name: encode_position(integer); Type: FUNCTION; Schema: public; Owner: poem
--

CREATE FUNCTION encode_position(pos integer) RETURNS text
    AS $$
 def encode_pos(positio):
   # Encodes number to string representation
   # 0: "0" = 0, "z" = 61
   # 1: "~0" = 62, "~z" = 123
   # 2: "~~00" = 124, "~~zz" = 3967
   # 3: "~~~000" = 3968, "~~~zzz" = 242295
```

```
   position = positio
   if position in range(0,10):
    return str(position)
   if position in range(10,36):
      return chr(position+55)
   if position in range(16,62):
       return chr(position+61)
   if position in range(62,124):
       return "~" + encode_pos(position-62)
   if position in range(124, 3968):
       return "~~" + encode_pos((position-124)/62) + encode_pos((position-124)%62)
   if position in range(3968, 242296):
    position -= 3968
    digit1 = position / (62 * 62)
    digit2 = (position % (62 * 62)) / 62
    digit3 = (position % (62 * 62)) % 62
    return "~~~" + encode_pos(digit1) + encode_pos(digit2) + encode_pos(digit3)
   raise "Stored Procedure: Encode Postition: Position out of range."

 return encode_pos(pos)
 $$
    LANGUAGE plpythonu IMMUTABLE;


ALTER FUNCTION public.encode_position(pos integer) OWNER TO poem;

--
-- Name: ensure_descendant(text, integer); Type: FUNCTION; Schema: public; Owner: poem
--

CREATE FUNCTION ensure_descendant(root_hierarchy text, target integer) RETURNS structure
    AS $$
         declare
           result "structure";
         begin
           select * into result
           from "structure"
           where hierarchy like root_hierarchy || '%'
                           and ident_id = target;

           if not found then
             insert             into             "structure"(hierarchy,             ident_id)
values(next_child_position(root_hierarchy), target) returning * into result;
           end if;

           return result;
         end;
       $$
    LANGUAGE plpgsql;


ALTER FUNCTION public.ensure_descendant(root_hierarchy text, target integer) OWNER TO poem;

--
-- Name: friend_dec_counter(integer, integer, integer); Type: FUNCTION; Schema: public; Owner:
poem
--

CREATE FUNCTION friend_dec_counter(subject_id1 integer, subject_id2 integer, count integer)
RETURNS void
    AS $$ DECLARE
       result friend;
BEGIN
       SELECT friend.* INTO result FROM friend WHERE
              (friend.subject_id=subject_id1 AND friend.friend_id=subject_id2)
              OR (friend.friend_id=subject_id1 AND friend.subject_id=subject_id2);
```

```
        IF FOUND AND result.model_count > count THEN
                UPDATE friend SET model_count=result.model_count - count
                WHERE friend.subject_id=result.subject_id
                AND friend.friend_id=result.friend_id;
        ELSE
                UPDATE friend SET model_count=0
                WHERE friend.subject_id=result.subject_id
                AND friend.friend_id=result.friend_id;
        END IF;
END;$$
    LANGUAGE plpgsql;


ALTER FUNCTION public.friend_dec_counter(subject_id1 integer, subject_id2 integer, count
integer) OWNER TO poem;

--
-- Name: friend_inc_counter(integer, integer, integer); Type: FUNCTION; Schema: public; Owner:
poem
--

CREATE FUNCTION friend_inc_counter(subject_id1 integer, subject_id2 integer, count integer)
RETURNS void
    AS $$ DECLARE
        result friend;
BEGIN
        PERFORM * FROM identity WHERE ((identity.id=subject_id2 OR identity.id=subject_id1) AND
identity.uri='public');
        IF (subject_id1=subject_id2 OR FOUND) THEN
                RETURN;
        END IF;

        SELECT friend.* INTO result FROM friend WHERE
                (friend.subject_id=subject_id1 AND friend.friend_id=subject_id2)
                OR (friend.friend_id=subject_id1 AND friend.subject_id=subject_id2);

        IF NOT FOUND THEN
                INSERT INTO friend (subject_id, friend_id, model_count)
                VALUES (subject_id1, subject_id2, 1);
        ELSE
                UPDATE friend SET model_count=result.model_count + count
                WHERE friend.subject_id=result.subject_id
                AND friend.friend_id=result.friend_id;
        END IF;
END;$$
    LANGUAGE plpgsql;


ALTER FUNCTION public.friend_inc_counter(subject_id1 integer, subject_id2 integer, count
integer) OWNER TO poem;

--
-- Name: friend_init(); Type: FUNCTION; Schema: public; Owner: poem
--

CREATE FUNCTION friend_init() RETURNS void
    AS $$ DECLARE
        user_pair record;
BEGIN
        DELETE FROM friend;
        PERFORM friend_inc_counter(get_identity_id_from_hierarchy(user1.subject),
                        get_identity_id_from_hierarchy(user2.subject), 1)
                FROM interaction as user1, interaction as user2
                WHERE user1.object=user2.object;
        -- halve the counter since all friends are counted twice
```

```
        UPDATE friend SET model_count=model_count / 2;
END;$$
    LANGUAGE plpgsql;


ALTER FUNCTION public.friend_init() OWNER TO poem;

--
-- Name: friend_trigger_interaction(); Type: FUNCTION; Schema: public; Owner: poem
--

CREATE OR REPLACE FUNCTION friend_trigger_interaction() RETURNS TRIGGER AS
$BODY$ DECLARE
        model_hierarchy text;
        user_hierarchy text;
        subject identity;
        friend_id integer;
BEGIN
        IF (TG_OP = 'INSERT') THEN
                model_hierarchy := NEW.object;
                user_hierarchy := NEW.subject;
        ELSEIF (TG_OP = 'DELETE') THEN
                model_hierarchy := OLD.object;
                user_hierarchy := OLD.subject;
        END IF;

        SELECT * INTO subject FROM get_identity_from_hierarchy(user_hierarchy);

        FOR friend_id IN SELECT access.subject_id FROM access, structure
                        WHERE              (access.object_id=structure.ident_id              AND
structure.hierarchy=model_hierarchy) LOOP

                IF (TG_OP = 'INSERT') THEN
                        PERFORM friend_inc_counter(subject.id, friend_id, 1);
                ELSEIF (TG_OP = 'DELETE') THEN
                        PERFORM friend_dec_counter(subject.id, friend_id, 1);
                END IF;
        END LOOP;
        RETURN NULL;
END;$BODY$
LANGUAGE 'plpgsql' VOLATILE
COST 100;

ALTER FUNCTION public.friend_trigger_interaction() OWNER TO poem;

--
-- Name: get_identity_from_hierarchy(text); Type: FUNCTION; Schema: public; Owner: poem
--

CREATE FUNCTION get_identity_from_hierarchy(hierarchy text) RETURNS identity
    AS $$ DECLARE
        result identity;
BEGIN
        SELECT     identity.*     INTO     result     FROM     identity,     structure     WHERE
identity.id=structure.ident_id AND structure.hierarchy=hierarchy;
        RETURN result;
END;$$
    LANGUAGE plpgsql;


ALTER FUNCTION public.get_identity_from_hierarchy(hierarchy text) OWNER TO poem;

--
-- Name: get_identity_id_from_hierarchy(text); Type: FUNCTION; Schema: public; Owner: poem
--
```

```
CREATE FUNCTION get_identity_id_from_hierarchy(hierarchy text) RETURNS integer
    AS $$ DECLARE
        result integer;
BEGIN
       SELECT    identity.id    INTO    result    FROM    identity,    structure    WHERE
identity.id=structure.ident_id AND structure.hierarchy=hierarchy;
       RETURN result;
END;$$
    LANGUAGE plpgsql;


ALTER FUNCTION public.get_identity_id_from_hierarchy(hierarchy text) OWNER TO poem;

--
-- Name: identity(text); Type: FUNCTION; Schema: public; Owner: poem
--

CREATE FUNCTION identity(openid text) RETURNS identity
    AS $$
         declare
           result "identity";
         begin
           select * into result
           from "identity"
           where uri = openid;

           if not found then
             insert into "identity"(uri) values(openid) returning * into result;
           end if;

           return result;
         end;
       $$
    LANGUAGE plpgsql;


ALTER FUNCTION public.identity(openid text) OWNER TO poem;

--
-- Name: is_shared(integer); Type: FUNCTION; Schema: public; Owner: poem
--

CREATE FUNCTION is_shared(id integer) RETURNS integer
    AS $$
          declare
            result integer;
          begin
            select count(*) into result
            from interaction, structure as subject, structure as object
            where                    interaction.object=object.hierarchy                    and
interaction.subject=subject.hierarchy and object.ident_id=id;

            return result;
          end;
        $$
    LANGUAGE plpgsql;


ALTER FUNCTION public.is_shared(id integer) OWNER TO poem;

--
-- Name: next_child_position(text); Type: FUNCTION; Schema: public; Owner: poem
--

CREATE FUNCTION next_child_position(hierarchy text) RETURNS text
    AS $_$
```

```
         select   $1   ||   encode_position(coalesce(max(child_position(hierarchy))+1,0))   from
"structure" where parent(hierarchy) = $1;
       $_$
    LANGUAGE sql;


ALTER FUNCTION public.next_child_position(hierarchy text) OWNER TO poem;

--
-- Name: poem_path(text); Type: FUNCTION; Schema: public; Owner: poem
--

CREATE FUNCTION poem_path(hierarchy text) RETURNS SETOF text
    AS $$
 # Returns an ordered collection of strings. The first item of the collection is
 # the path to the first item in the hierarchy and so on.
 all = ["",""]
 if (hierarchy == ""):
        return all
 else:
        position = 0
        # for each character in the input string
        while position < len(hierarchy):
                # count tildes
                tilde_count = 0
                while hierarchy[position+tilde_count] == "~":
                        tilde_count += 1
                if tilde_count == 0:
                        all.append(hierarchy[:position+1])
                        position += 1
                else:
                        all.append(hierarchy[:position+2*tilde_count])
                position += 2*tilde_count;
        return all
 $$
    LANGUAGE plpythonu IMMUTABLE;


ALTER FUNCTION public.poem_path(hierarchy text) OWNER TO poem;

--
-- Name: work_around_path(text); Type: FUNCTION; Schema: public; Owner: poem
--

CREATE FUNCTION work_around_path(hierarchy text) RETURNS SETOF text
    AS $$
 def poem_path(hierarchy):
        # Returns an ordered collection of strings. The first item of the collection is
        # the path to the first item in the hierarchy and so on.
        all = ["",""]
        if (hierarchy == ""):
                return all
        else:
                position = 0
                # for each character in the input string
                while position < len(hierarchy):
                        # count tildes
                        tilde_count = 0
                        while hierarchy[position+tilde_count] == "~":
                                tilde_count += 1
                        if tilde_count == 0:
                                all.append(hierarchy[:position+1])
                                position += 1
                        else:
                                all.append(hierarchy[:position+2*tilde_count])
                        position += 2*tilde_count;
```

```
              return all
   return poem_path(hierarchy)[2:-1]
 $$
    LANGUAGE plpythonu IMMUTABLE;


ALTER FUNCTION public.work_around_path(hierarchy text) OWNER TO poem;

--
-- Name: content_id_seq; Type: SEQUENCE; Schema: public; Owner: poem
--

CREATE SEQUENCE content_id_seq
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;


ALTER TABLE public.content_id_seq OWNER TO poem;

--
-- Name: content_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: poem
--

ALTER SEQUENCE content_id_seq OWNED BY content.id;


--
-- Name: content_id_seq; Type: SEQUENCE SET; Schema: public; Owner: poem
--

SELECT pg_catalog.setval('content_id_seq', 1, false);


--
-- Name: friend_id_seq; Type: SEQUENCE; Schema: public; Owner: poem
--

CREATE SEQUENCE friend_id_seq
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;


ALTER TABLE public.friend_id_seq OWNER TO poem;

--
-- Name: friend_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: poem
--

ALTER SEQUENCE friend_id_seq OWNED BY friend.id;


--
-- Name: friend_id_seq; Type: SEQUENCE SET; Schema: public; Owner: poem
--

SELECT pg_catalog.setval('friend_id_seq', 1, false);


--
-- Name: identity_id_seq; Type: SEQUENCE; Schema: public; Owner: poem
```

```
--

CREATE SEQUENCE identity_id_seq
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;


ALTER TABLE public.identity_id_seq OWNER TO poem;

--
-- Name: identity_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: poem
--

ALTER SEQUENCE identity_id_seq OWNED BY identity.id;


--
-- Name: identity_id_seq; Type: SEQUENCE SET; Schema: public; Owner: poem
--

SELECT pg_catalog.setval('identity_id_seq', 5, true);


--
-- Name: interaction_id_seq; Type: SEQUENCE; Schema: public; Owner: poem
--

CREATE SEQUENCE interaction_id_seq
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;


ALTER TABLE public.interaction_id_seq OWNER TO poem;

--
-- Name: interaction_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: poem
--

ALTER SEQUENCE interaction_id_seq OWNED BY interaction.id;


--
-- Name: interaction_id_seq; Type: SEQUENCE SET; Schema: public; Owner: poem
--

SELECT pg_catalog.setval('interaction_id_seq', 1, true);


--
-- Name: model_rating_id_seq; Type: SEQUENCE; Schema: public; Owner: poem
--

CREATE SEQUENCE model_rating_id_seq
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;


ALTER TABLE public.model_rating_id_seq OWNER TO poem;
```

```
--
-- Name: model_rating_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: poem
--

ALTER SEQUENCE model_rating_id_seq OWNED BY model_rating.id;


--
-- Name: model_rating_id_seq; Type: SEQUENCE SET; Schema: public; Owner: poem
--

SELECT pg_catalog.setval('model_rating_id_seq', 1, false);


--
-- Name: representation_id_seq; Type: SEQUENCE; Schema: public; Owner: poem
--

CREATE SEQUENCE representation_id_seq
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;


ALTER TABLE public.representation_id_seq OWNER TO poem;

--
-- Name: representation_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: poem
--

ALTER SEQUENCE representation_id_seq OWNED BY representation.id;


--
-- Name: representation_id_seq; Type: SEQUENCE SET; Schema: public; Owner: poem
--

SELECT pg_catalog.setval('representation_id_seq', 1, true);


--
-- Name: id; Type: DEFAULT; Schema: public; Owner: poem
--

ALTER TABLE content ALTER COLUMN id SET DEFAULT nextval('content_id_seq'::regclass);


--
-- Name: id; Type: DEFAULT; Schema: public; Owner: poem
--

ALTER TABLE friend ALTER COLUMN id SET DEFAULT nextval('friend_id_seq'::regclass);


--
-- Name: id; Type: DEFAULT; Schema: public; Owner: poem
--

ALTER TABLE identity ALTER COLUMN id SET DEFAULT nextval('identity_id_seq'::regclass);


--
-- Name: id; Type: DEFAULT; Schema: public; Owner: poem
--
```

```
ALTER TABLE interaction ALTER COLUMN id SET DEFAULT nextval('interaction_id_seq'::regclass);


--
-- Name: id; Type: DEFAULT; Schema: public; Owner: poem
--

ALTER TABLE model_rating ALTER COLUMN id SET DEFAULT nextval('model_rating_id_seq'::regclass);


--
-- Name: id; Type: DEFAULT; Schema: public; Owner: poem
--

ALTER TABLE representation ALTER COLUMN id SET DEFAULT
nextval('representation_id_seq'::regclass);


--
-- Data for Name: comment; Type: TABLE DATA; Schema: public; Owner: poem
--

COPY comment (id, subject_id, title, content) FROM stdin;
\.


--
-- Data for Name: content; Type: TABLE DATA; Schema: public; Owner: poem
--

COPY content (id, erdf, svg, png_large, png_small) FROM stdin;
\.


--
-- Data for Name: friend; Type: TABLE DATA; Schema: public; Owner: poem
--

COPY friend (id, subject_id, friend_id, model_count) FROM stdin;
\.


--
-- Data for Name: identity; Type: TABLE DATA; Schema: public; Owner: poem
--

COPY identity (id, uri) FROM stdin;
0
1       root
2       public
3       groups
4       ownership
\.


--
-- Data for Name: interaction; Type: TABLE DATA; Schema: public; Owner: poem
--

COPY interaction (id, subject, subject_descend, object, object_self, object_descend,
object_restrict_to_parent, scheme, term) FROM stdin;
1       U2      t       U2      f       t       t       http://b3mn.org/http    owner
\.


--
```

```
-- Data for Name: model_rating; Type: TABLE DATA; Schema: public; Owner: poem
--

COPY model_rating (id, subject_id, object_id, score) FROM stdin;
\.


--
-- Data for Name: plugin; Type: TABLE DATA; Schema: public; Owner: poem
--

COPY plugin (rel, title, description, java_class, is_export) FROM stdin;
/self   ModelHandler   Open model in the editor      org.b3mn.poem.handler.ModelHandler    t
/repository    RepositoryHandler     Returns          the          Repository          base
       org.b3mn.poem.handler.RepositoryHandler      f
/model_types   ModelHandler   Open           model          in          the          editor
       org.b3mn.poem.handler.ModelHandler    f
/model  CollectionHandler     Open           model          in          the          editor
       org.b3mn.poem.handler.CollectionHandler      f
/new    NewModelHandler       Open           model          in          the          editor
       org.b3mn.poem.handler.NewModelHandler f
/info   InfoHandler    edit info      org.b3mn.poem.handler.InfoHandler     f
/access AccessHandler  edit access    org.b3mn.poem.handler.AccessHandler   f
/info-access   MetaHandler    About   org.b3mn.poem.handler.MetaHandler     t
/svg    ImageRenderer  Model as SVG   org.b3mn.poem.handler.ImageRenderer   t
/pdf    PdfRenderer    Model as PDF   org.b3mn.poem.handler.PdfRenderer     t
/png    PngRenderer    Model as PNG   org.b3mn.poem.handler.PngRenderer     t
/rdf    RdfExporter    Model as RDF   org.b3mn.poem.handler.RdfExporter     t
/login  OpenID 2.0 Login      Handles the login and stores OpenID attributes in the database
       org.b3mn.poem.handler.LoginHandler    f
/user   UserHandler    Manages user meta data org.b3mn.poem.handler.UserHandler     f
/repository2   Repository RELOADED   Returns  initial  Html  page  for  the  new  repository
       org.b3mn.poem.handler.Repository2Handler     f
/meta   Model Info Handler    Handles Requests from Repository2 concerning object meta data
       org.b3mn.poem.handler.ModelInfoHandler       f
/tags   Tag Handler    Handles      all      requests      concerning      model      tagging
       org.b3mn.poem.handler.TagHandler      f
/filter Filter Handler Handles       client       request       for       server       filter
       org.b3mn.poem.handler.SortFilterHandler      f
\.


--
-- Data for Name: representation; Type: TABLE DATA; Schema: public; Owner: poem
--

COPY representation (id, ident_id, mime_type, language, title, summary, created, updated, type)
FROM stdin;
\.


--
-- Data for Name: schema_info; Type: TABLE DATA; Schema: public; Owner: poem
--

COPY schema_info (version) FROM stdin;
5
\.


--
-- Data for Name: setting; Type: TABLE DATA; Schema: public; Owner: poem
--

COPY setting (subject_id, id, key, value) FROM stdin;
0       1       UserManager.DefaultCountryCode       us
```

```
0       2       UserManager.DefaultLanguageCode       en
\.


--
-- Data for Name: structure; Type: TABLE DATA; Schema: public; Owner: poem
--

COPY structure (hierarchy, ident_id) FROM stdin;
U       1
U1      2
U2      4
U3      3
\.


--
-- Data for Name: subject; Type: TABLE DATA; Schema: public; Owner: poem
--

COPY subject (ident_id, nickname, email, fullname, dob, gender, postcode, first_login,
last_login, login_count, language_code, country_code, password, visibility) FROM stdin;
2       \N      \N      \N      \N      \N      \N      2008-01-01      2008-01-01      0       \N
        \N      \N      \N
\.


--
-- Data for Name: tag_definition; Type: TABLE DATA; Schema: public; Owner: poem
--

COPY tag_definition (id, subject_id, name) FROM stdin;
\.


--
-- Data for Name: tag_relation; Type: TABLE DATA; Schema: public; Owner: poem
--

COPY tag_relation (id, tag_id, object_id) FROM stdin;
\.


--
-- Name: comment_pkey; Type: CONSTRAINT; Schema: public; Owner: poem; Tablespace:
--

ALTER TABLE ONLY comment
    ADD CONSTRAINT comment_pkey PRIMARY KEY (id);


--
-- Name: content_pkey; Type: CONSTRAINT; Schema: public; Owner: poem; Tablespace:
--

ALTER TABLE ONLY content
    ADD CONSTRAINT content_pkey PRIMARY KEY (id);


--
-- Name: friend_pkey; Type: CONSTRAINT; Schema: public; Owner: poem; Tablespace:
--

ALTER TABLE ONLY friend
    ADD CONSTRAINT friend_pkey PRIMARY KEY (id);
```

```
--
-- Name: identity_pkey; Type: CONSTRAINT; Schema: public; Owner: poem; Tablespace:
--

ALTER TABLE ONLY identity
    ADD CONSTRAINT identity_pkey PRIMARY KEY (id);


--
-- Name: interaction_pkey; Type: CONSTRAINT; Schema: public; Owner: poem; Tablespace:
--

ALTER TABLE ONLY interaction
    ADD CONSTRAINT interaction_pkey PRIMARY KEY (id);


--
-- Name: model_rating_pkey; Type: CONSTRAINT; Schema: public; Owner: poem; Tablespace:
--

ALTER TABLE ONLY model_rating
    ADD CONSTRAINT model_rating_pkey PRIMARY KEY (id);


--
-- Name: representation_pkey; Type: CONSTRAINT; Schema: public; Owner: poem; Tablespace:
--

ALTER TABLE ONLY representation
    ADD CONSTRAINT representation_pkey PRIMARY KEY (id);


--
-- Name: settings_pkey; Type: CONSTRAINT; Schema: public; Owner: poem; Tablespace:
--

ALTER TABLE ONLY setting
    ADD CONSTRAINT settings_pkey PRIMARY KEY (id);


--
-- Name: structure_pkey; Type: CONSTRAINT; Schema: public; Owner: poem; Tablespace:
--

ALTER TABLE ONLY structure
    ADD CONSTRAINT structure_pkey PRIMARY KEY (hierarchy);


--
-- Name: tag_definition_pkey; Type: CONSTRAINT; Schema: public; Owner: poem; Tablespace:
--

ALTER TABLE ONLY tag_definition
    ADD CONSTRAINT tag_definition_pkey PRIMARY KEY (id);


--
-- Name: tag_relation_pkey; Type: CONSTRAINT; Schema: public; Owner: poem; Tablespace:
--

ALTER TABLE ONLY tag_relation
    ADD CONSTRAINT tag_relation_pkey PRIMARY KEY (id);


--
```

```
-- Name: user_pkey; Type: CONSTRAINT; Schema: public; Owner: poem; Tablespace:
--

ALTER TABLE ONLY subject
    ADD CONSTRAINT user_pkey PRIMARY KEY (ident_id);


--
-- Name: friend_identity_idx; Type: INDEX; Schema: public; Owner: poem; Tablespace:
--

CREATE INDEX friend_identity_idx ON friend USING btree (subject_id, friend_id);


--
-- Name: object_idx; Type: INDEX; Schema: public; Owner: poem; Tablespace:
--

CREATE INDEX object_idx ON interaction USING btree (object);


--
-- Name: poem_position; Type: INDEX; Schema: public; Owner: poem; Tablespace:
--

CREATE    INDEX    poem_position    ON    structure    USING    btree    (parent(hierarchy),
child_position(hierarchy));


--
-- Name: rel_idx; Type: INDEX; Schema: public; Owner: poem; Tablespace:
--

CREATE INDEX rel_idx ON plugin USING btree (rel);


--
-- Name: subject_idx; Type: INDEX; Schema: public; Owner: poem; Tablespace:
--

CREATE INDEX subject_idx ON interaction USING btree (subject);


--
-- Name: friend_interaction; Type: TRIGGER; Schema: public; Owner: poem
--

CREATE TRIGGER friend_interaction
    AFTER INSERT OR DELETE ON interaction
    FOR EACH ROW
    EXECUTE PROCEDURE friend_trigger_interaction();


--
-- Name: comment_identity_fkey; Type: FK CONSTRAINT; Schema: public; Owner: poem
--

ALTER TABLE ONLY comment
    ADD CONSTRAINT comment_identity_fkey FOREIGN KEY (subject_id) REFERENCES identity(id);


--
-- Name: content_id_fkey; Type: FK CONSTRAINT; Schema: public; Owner: poem
--

ALTER TABLE ONLY content
```

```
    ADD CONSTRAINT content_id_fkey FOREIGN KEY (id) REFERENCES representation(id) ON DELETE
CASCADE;


--
-- Name: fkey_friend_identity2; Type: FK CONSTRAINT; Schema: public; Owner: poem
--

ALTER TABLE ONLY friend
    ADD CONSTRAINT fkey_friend_identity2 FOREIGN KEY (friend_id) REFERENCES identity(id) ON
DELETE CASCADE;


--
-- Name: friends_fkey_identity1; Type: FK CONSTRAINT; Schema: public; Owner: poem
--

ALTER TABLE ONLY friend
    ADD CONSTRAINT friends_fkey_identity1 FOREIGN KEY (subject_id) REFERENCES identity(id) ON
DELETE CASCADE;


--
-- Name: interaction_object_fkey; Type: FK CONSTRAINT; Schema: public; Owner: poem
--

ALTER TABLE ONLY interaction
    ADD CONSTRAINT interaction_object_fkey FOREIGN KEY (object) REFERENCES structure(hierarchy)
ON DELETE CASCADE;


--
-- Name: interaction_subject_fkey; Type: FK CONSTRAINT; Schema: public; Owner: poem
--

ALTER TABLE ONLY interaction
    ADD    CONSTRAINT    interaction_subject_fkey    FOREIGN    KEY    (subject)    REFERENCES
structure(hierarchy) ON DELETE CASCADE;


--
-- Name: model_rating_object_fkey; Type: FK CONSTRAINT; Schema: public; Owner: poem
--

ALTER TABLE ONLY model_rating
    ADD CONSTRAINT model_rating_object_fkey FOREIGN KEY (object_id) REFERENCES identity(id) ON
DELETE CASCADE;


--
-- Name: model_rating_subject_fkey; Type: FK CONSTRAINT; Schema: public; Owner: poem
--

ALTER TABLE ONLY model_rating
    ADD CONSTRAINT model_rating_subject_fkey FOREIGN KEY (subject_id) REFERENCES identity(id)
ON DELETE CASCADE;


--
-- Name: representation_ident_id_fkey; Type: FK CONSTRAINT; Schema: public; Owner: poem
--

ALTER TABLE ONLY representation
    ADD CONSTRAINT representation_ident_id_fkey FOREIGN KEY (ident_id) REFERENCES identity(id)
ON DELETE CASCADE;
```

```
--
-- Name: structure_ident_id_fkey; Type: FK CONSTRAINT; Schema: public; Owner: poem
--

ALTER TABLE ONLY structure
    ADD CONSTRAINT structure_ident_id_fkey FOREIGN KEY (ident_id) REFERENCES identity(id) ON
DELETE CASCADE;


--
-- Name: subject_id_fkey; Type: FK CONSTRAINT; Schema: public; Owner: poem
--

ALTER TABLE ONLY setting
    ADD CONSTRAINT subject_id_fkey FOREIGN KEY (subject_id) REFERENCES identity(id) ON DELETE
CASCADE;


--
-- Name: tag_definition_subject_id_fkey; Type: FK CONSTRAINT; Schema: public; Owner: poem
--

ALTER TABLE ONLY tag_definition
    ADD  CONSTRAINT  tag_definition_subject_id_fkey  FOREIGN  KEY  (subject_id)  REFERENCES
identity(id) ON DELETE CASCADE;


--
-- Name: tag_relation_identity; Type: FK CONSTRAINT; Schema: public; Owner: poem
--

ALTER TABLE ONLY tag_relation
    ADD CONSTRAINT tag_relation_identity FOREIGN KEY (object_id) REFERENCES identity(id) ON
DELETE CASCADE;


--
-- Name: tag_relation_tag_pkey; Type: FK CONSTRAINT; Schema: public; Owner: poem
--

ALTER TABLE ONLY tag_relation
    ADD CONSTRAINT tag_relation_tag_pkey FOREIGN KEY (tag_id) REFERENCES tag_definition(id) ON
DELETE CASCADE;


--
-- Name: user_ident_id_pkey; Type: FK CONSTRAINT; Schema: public; Owner: poem
--

ALTER TABLE ONLY subject
    ADD CONSTRAINT user_ident_id_pkey FOREIGN KEY (ident_id) REFERENCES identity(id) ON DELETE
CASCADE;


--
-- Name: public; Type: ACL; Schema: -; Owner: postgres
--

REVOKE ALL ON SCHEMA public FROM PUBLIC;
REVOKE ALL ON SCHEMA public FROM postgres;
GRANT ALL ON SCHEMA public TO postgres;
GRANT ALL ON SCHEMA public TO PUBLIC;


--
```

```
-- PostgreSQL database dump complete
--
```

# ANNEXE V

## Oryx – résultat de l'exécution du script db_schema.sql

```
D:\workspace\Oryx\oryx\poem-jvm\data\database>psql poem < db_schema.sql postgres
SET
SET
SET
SET
SET
ERROR:  language "plpgsql" already exists
ALTER LANGUAGE
CREATE LANGUAGE
ALTER LANGUAGE
SET
CREATE FUNCTION
ALTER FUNCTION
CREATE FUNCTION
ALTER FUNCTION
SET
SET
CREATE TABLE
ALTER TABLE
ALTER TABLE
CREATE TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
CREATE TABLE
ALTER TABLE
ALTER TABLE
CREATE VIEW
ALTER TABLE
CREATE SEQUENCE
ALTER TABLE
 setval
--------
      1
(1 ligne)

CREATE TABLE
ALTER TABLE
CREATE TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
CREATE TABLE
ALTER TABLE
CREATE TABLE
ALTER TABLE
CREATE TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
CREATE TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
```

```
ALTER TABLE
ALTER TABLE
CREATE TABLE
ALTER TABLE
CREATE SEQUENCE
ALTER TABLE
 setval
--------
       3
(1 ligne)

CREATE TABLE
ALTER TABLE
CREATE TABLE
ALTER TABLE
CREATE SEQUENCE
ALTER TABLE
 setval
--------
       1
(1 ligne)

CREATE TABLE
ALTER TABLE
CREATE SEQUENCE
ALTER TABLE
 setval
--------
       1
(1 ligne)

CREATE TABLE
ALTER TABLE
CREATE FUNCTION
ALTER FUNCTION
CREATE FUNCTION
ALTER FUNCTION
CREATE FUNCTION
ALTER FUNCTION
CREATE FUNCTION
ALTER FUNCTION
CREATE FUNCTION
ALTER FUNCTION
CREATE FUNCTION
ALTER FUNCTION
CREATE FUNCTION
ALTER FUNCTION
CREATE FUNCTION
ALTER FUNCTION
CREATE FUNCTION
ALTER FUNCTION
CREATE FUNCTION
ALTER FUNCTION
CREATE FUNCTION
ALTER FUNCTION
CREATE FUNCTION
ALTER FUNCTION
CREATE FUNCTION
ALTER FUNCTION
CREATE FUNCTION
ALTER FUNCTION
CREATE FUNCTION
ALTER FUNCTION
CREATE SEQUENCE
ALTER TABLE
ALTER SEQUENCE
```

```
 setval
--------
       1
(1 ligne)

CREATE SEQUENCE
ALTER TABLE
ALTER SEQUENCE
 setval
--------
       1
(1 ligne)

CREATE SEQUENCE
ALTER TABLE
ALTER SEQUENCE
 setval
--------
       5
(1 ligne)

CREATE SEQUENCE
ALTER TABLE
ALTER SEQUENCE
 setval
--------
       1
(1 ligne)

CREATE SEQUENCE
ALTER TABLE
ALTER SEQUENCE
 setval
--------
       1
(1 ligne)

CREATE SEQUENCE
ALTER TABLE
ALTER SEQUENCE
 setval
--------
       1
(1 ligne)

ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
CREATE INDEX
CREATE INDEX
CREATE INDEX
```

```
CREATE INDEX
CREATE INDEX
CREATE TRIGGER
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
REVOKE
REVOKE
GRANT
GRANT

D:\workspace\Oryx\oryx\poem-
```

# LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES

## Travaux cités


Abbes, M., Khomh, F., Guéhéneuc, Y.-G., & Antoniol, G. (2011). An empirical study of the impact of two antipatterns, blob and spaghetti code, on program comprehension. *Software Maintenance and Reengineering (CSMR), 2011 15th European Conference on*, (pp. 181--190).

Acher, M., Cleve, A., Collet, P., Merle, P., Duchien, L., & Lahire, P. (2011). Reverse engineering architectural feature models. Springer.

Adams, B., Tromp, H., De Schutter, K., & De Meuter, W. (2007). Design recovery and maintenance of build systems. *Software Maintenance, 2007. ICSM 2007. IEEE International Conference on*, (pp. 114--123).

Akkiraju, R., Mitra, T., & Thulasiram, U. (s.d.). Reverse engineering platform independent models from business software applications. *chapter, 4*, 83--94.

Alex Woodie. (2014, 01 27). *When to Hadoop, and when not to*. (datanami) Consulté le 12 05, 2014, sur http://www.datanami.com/2014/01/27/when_to_hadoop_and_when_not_to/

Alnusair, A., & Zhao, T. (2010). Using Semantic Inference for Software Understanding and Design Recovery. *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*, (pp. 980--985).

Ami, T., & Sommer, R. (2007). Comparison and evaluation of business process modelling and management tools. *International Journal of Services and Standards, 3*(2), 249--261.

Apache Foundation. (s.d.). *Apache Hadoop*. (The Apache Software Foundation) Consulté le 10 15, 2015, sur http://hadoop.apache.org

Apache Foundation. (s.d.). *Apache HBase*. (The Apache Software Foundation) Consulté le 10 15, 2015, sur https://hbase.apache.org

Apache Foundation. (s.d.). *Apache sqoop*. (The Apache Software Foundation) Consulté le 10 15, 2015, sur sqoop.apache.org

Apache Foundation. (s.d.). *HBase on Windows*. (The Apache Software Foundation) Consulté le 10 15, 2015, sur https://hbase.apache.org/cygwin.html

Austin, M., & Samadzadeh, M. (2005). Software comprehension/maintenance: An introductory course. *Systems Engineering, 2005. ICSEng 2005. 18th International Conference on*, (pp. 414--419).

Begoli, E. (2015, 04 14). *3 Reasons For Why Should You Use Hibernate*. Récupéré sur Toolbox.com: http://it.toolbox.com/blogs/lim/3-reasons-for-why-should-you-use-hibernate-3863

Belghait, F. (2012). Exploration de la migration de la base de données relationnelle du système de gestion de processus d'affaires Oryx vers la base de données NO-SQL utilisée par la plateforme de l'informatique de nuage de Hadoop.

Bergmann, S. (2007). Design and Implementation of a Workflow Engine.

Bernard, E. (s.d.). *Hibernate OGM configuration before / after*. Consulté le 09 15, 2015, sur https://gist.github.com/emmanuelbernard/1044781

Booch, G. (2008). Software Archeology and the Handbook of Software Architecture. *Workshop Software Reengineering*, *126*, pp. 5--6.

Canfora, G., Di Penta, M., & Cerulo, L. (2011). Achievements and challenges in software reverse engineering. *Communications of the ACM, 54*(4), 142--151.

Chen, M., Mao, S., Zhang, Y., & Leung, V. (2014). Big data: related technologies, challenges and future prospects. Springer.

Cloudera. (s.d.). *Quickstart Tutorial.* Consulté le 10 15, 2015, sur http://quickstart.cloudera/#/tutorial

Cloudera. (s.d.). *The Platform for Big Data and the Leading Solution for Apache Hadoop in the Enterprise - Cloudera*. (Cloudera) Consulté le 09 22, 2015, sur http://www.cloudera.com

Collard, M. (2005). Addressing source code using srcml. *IEEE International Workshop on Program Comprehension Working Session: Textual Views of Source Code to Support Comprehension (IWPC'05).*

Constantinou, E., Kakarontzas, G., & Stamelos, I. (2011). Towards open source software system architecture recovery using design metrics. *Informatics (PCI), 2011 15th Panhellenic Conference on*, (pp. 166--170).

Cosma, D. (2010). Reverse engineering object-oriented distributed systems. *Software Maintenance (ICSM), 2010 IEEE International Conference on*, (pp. 1--6).

Cryans, J.-D., April, A., & Abran, A. (2008). Criteria to compare cloud computing with current database technology. Springer.

Cubranic, D., & Murphy, G. (2003). Hipikat: Recommending pertinent software development artifacts. *Software Engineering, 2003. Proceedings. 25th International Conference on*, (pp. 408--418).

Davies, M. (2013). Data-Driven BPM: Making "Big Data" Actionable. *http://smartdatacollective.com/node/104576*.

De Carlini, U., De Lucia, A., Di Lucca, G., & Tortora, G. (1993). An integrated and interactive reverse engineering environment for existing software comprehension. *Program Comprehension, 1993. Proceedings., IEEE Second Workshop on*, (pp. 128--137).

Decker, G., Overdick, H., & Weske, M. (2008). Oryx--an open modeling platform for the BPM community. 382--385. Springer.

Deimel, L., & Naveda, J. (1990). *Reading computer programs: Instructor's guide to exercises.*

Demeyer, S., Ducasse, S., & Nierstrasz, O. (2002). *Object-oriented reengineering patterns.* Elsevier.

Dugerdil, P., & Repond, J. (2010). Automatic generation of abstract views for legacy software comprehension. *Proceedings of the 3rd India software engineering conference*, (pp. 23--32).

Dumas, M., La Rosa, M., Mendling, J., & Reijers, H. (2013). *Fundamentals of business process management.* Springer.

Dunning, T., & Friedman, E. (2015). *Real-World Hadoop.* O'Reilly.

El Beggar Omar, B., & Taoufiq, G. (2013). Comparative Study between Clustering and Model Driven Reverse Engineering Approaches. *Lecture Notes on Software Engineering, 1*(2).

Emailed, E. J. (2010, 01 17). *Hibernate Vs. JDBC ( A comparison)*. Récupéré sur Java-Samples.com: http://www.java-samples.com/showtutorial.php?tutorialid=813

Faura, M. V. (2012). Making Big Data Useful with BPM. *Enterprise Systems*.

Fei, X. (s.d.). *Java Example Code using HBase Data Model Operations*. (Xu Fei's Blog) Consulté le 10 15, 2015, sur https://autofei.wordpress.com/2012/04/02/java-example-code-using-hbase-data-model-operations/

Fishleigh, J. (2014). A Non-Technical Journey into the World of Big Data: an Introduction. *14(02)*, 149--151. Cambridge Univ Press.

Fowler, M. (s.d.). *Introduction to NoSQL*. (Youtube) Consulté le 10 15, 2015, sur https://www.youtube.com/watch?v=qI_g07C_Q5I

Gao, X. (2013). Towards the Next Generation Intelligent BPM--In the Era of Big Data. 4--9. Springer.

ggraham412. (2015, 09 03). *Apache Hbase Example Using Java*. (Code Project) Consulté le 10 15, 2015, sur http://www.codeproject.com/Articles/1024851/Apache-HBase-Example-Using-Java

Gilbert, P. (2005). *What is the difference between Workflow Engines and BPM Suites.*

Gordon, K. (2013). What is Big Data? *55(3)*, 12--13. Br Computer Soc.

*Hadoop Administrator Path*. (s.d.). (Cloudera) Consulté le 09 04, 2015, sur http://fr.cloudera.com/content/cloudera/en/training/roles/administrators.html#

*Hadoop Data Analyst Path*. (s.d.). (Cloudera) Consulté le 09 04, 2015, sur http://fr.cloudera.com/content/cloudera/en/training/roles/analysts.html

*Hadoop Developer Path*. (s.d.). (Cloudera) Consulté le 09 03, 2015, sur http://fr.cloudera.com/content/cloudera/en/training/roles/developers.html

Hassan, A., & Holt, R. (2004). Using development history sticky notes to understand software architecture. *Program Comprehension, 2004. Proceedings. 12th IEEE International Workshop on*, (pp. 183--192).

Hibernate OGM. (2015, 11 03). *Hibernate OGM*. Récupéré sur Hibernate OGM: http://hibernate.org/ogm/

Hibernate ORM. (2015, 11 03). *Hibernate ORM*. Récupéré sur Hibernate. Everything Data.: http://hibernate.org/orm/

Hightower, R. (s.d.). *Hibernate Object Mapping for NoSQL Datastores*. (InfoQ) Consulté le 10 25, 2015, sur http://www.infoq.com/news/2011/07/hibernateogm

Hill, E. (2008). Developing natural language-based program analyses and tools to expedite software maintenance. *Companion of the 30th international conference on Software engineering*, (pp. 1015--1018).

Hill, J. (2013). BPM Outlook: Gartner Outlines 2013 Trends.

HortonWorks. (s.d.). *Bienvenue à HortonWorks*. (HortonWorks) Consulté le 10 15, 2015, sur https://fr.hortonworks.com/

Hunt, A., & Thomas, D. (2002). Software archaeology. *Software, IEEE, 19*(2), 20--22.

IEEE Computer Society. (2014). *SWEBOK Guide to the Software Engineering Body of Knowledge.* IEEE.

Jahnke, J., Wadsack, J., & Zundorf, A. (2002). A history concept for design recovery tools. *Software Maintenance and Reengineering, 2002. Proceedings. Sixth European Conference on*, (pp. 37--46).

Jamal, S., & Favre, J.-M. (s.d.). Outils pour l'analyse de l'évolution des logiciels. *ICSSEA 2001, Paris, France. 4-6 Décembre 2001 Année 2002*.

Java Performance Tuning. (2015, 11 02). *The Interview: Gavin King, Hibernate*. Récupéré sur Newsletter 041: http://www.javaperformancetuning.com/news/interview041.shtml

Jin, D., & Cordy, J. (2006). Integrating reverse engineering tools using a service-sharing methodology. *Program Comprehension, 2006. ICPC 2006. 14th IEEE International Conference on*, (pp. 94--99).

Kanellopoulos, Y., & Tjortjis, C. (2004). Data mining source code to facilitate program comprehension: experiments on clustering data retrieved from C++ programs. *Program Comprehension, 2004. Proceedings. 12th IEEE International Workshop on*, (pp. 214--223).

Kazman, R., & Carrière, S. (1999). Playing detective: Reconstructing software architecture from available evidence. *Automated Software Engineering, 6*(2), 107--138.

Kienle, H., & Müller, H. (2010). Rigi—An environment for software reverse engineering, exploration, visualization, and redocumentation. *Science of Computer Programming, 75*(4), 247--263.

Kim, Y. M. (2008, 12 04). *Why i choose Hibernate for my project?* Récupéré sur mkyong.com: http://www.mkyong.com/hibernate/why-i-choose-hibernate-for-my-project/

Ko, R. (2009). A computer scientist's introductory guide to business process management (BPM). *Crossroads, 15*(4), 4.

Ko, R., Lee, S., & Lee, E. (2009). Business process management (BPM) standards: a survey. *Business Process Management Journal, 15*(5), 744--791.

Kruchten, P. (1995). The 4+ 1 view model of architecture. *Software, IEEE, 12*(6), 42--50.

Kumar, A. (2013). A study of the influence of code-tracing problems on code-writing skills. *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*, (pp. 183--188).

Kumar, N. (2013). An Approach for Reverse Engineering thorough Complementary Software Views.

Kunze, M., Luebbe, A., Weidlich, M., & Weske, M. (2011). Towards understanding process modeling--the case of the BPM academic initiative. 44--58. Springer.

Kuzmin, R. (2010, 12 05). *Why I do hate Hibernate*. Récupéré sur IT Roman: https://itroman.wordpress.com/2010/12/05/why-i-do-hate-hibernate/

Langit, L. (s.d.). *Hadoop MarReduce Fundamentals*. (Youtube) Consulté le 10 15, 2015, sur https://www.youtube.com/watch?v=7FcMhTTG1Cs

LAROCHELLE, H. (2011). EXPLORATION DE HADOOP ET HBASE PERSISTANCE DES DONNÉES D'UNE APPLICATION WEB AVEC UN MODÈLE NON RELATIONNEL.

Lauzon, D. (2012). Introduction to Big Data.

Lopez, M., Whalley, J., Robbins, P., & Lister, R. (2008). Relationships between reading, tracing and writing skills in introductory programming. *Proceedings of the Fourth international Workshop on Computing Education Research*, (pp. 101--112).

M.M. Lehman, J. R. (1997). *Metrics and Laws of Software Evolution.*

Malton, A., Schneider, K., Cordy, J., Dean, T., Cousineau, D., & Reynolds, J. (2001). Processing software source text in automated design recovery and transformation. *Program Comprehension, 2001. IWPC 2001. Proceedings. 9th International Workshop on*, (pp. 127--134).

Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., & Byers, A. (2011). Big data: The next frontier for innovation, competition, and productivity. *McKinsey Global Institute*, 1--137.

Mendling, J., Reijers, H., & van der Aalst, W. (2010). Seven process modeling guidelines (7PMG). *52(2)*, 127--136. Elsevier.

Moha, N., Gueheneuc, Y.-G., Duchien, L., & Le Meur, A. (2010). DECOR: A method for the specification and detection of code and design smells. *Software Engineering, IEEE Transactions on, 36*(1), 20--36.

Monsalve, C. (2012). Representation of business processes at multiple levels of abstraction (strategic, tactical and operational) during the requirements elicitation stage of a software project, and the measurement of their functional size with iso 19761.

Nowak, A. (s.d.). *Oryx-Editor Setup Development Environment*. Consulté le 05 08, 2015, sur https://code.google.com/p/oryx-editor/wiki/SetupDevelopmentEnvironment

O'Brien, M. (2003). Software comprehension--a review \& research direction. *Department of Computer Science \& Information Systems University of Limerick, Ireland, Technical Report*.

Office québécois de la langue française. (2015). *Le grand dictionnaire terminologique*. Consulté le 04 03, 2015, sur http://www.granddictionnaire.com/ficheOqlf.aspx?Id_Fiche=26501384

Oracle. (2015, 11 03). *Java SE Technologies - Database*. Récupéré sur www.oracle.com: http://www.oracle.com/technetwork/java/javase/jdbc/index.html

Oryx-Editor. (s.d.). *Downloads - Bachelor's papers*. Consulté le 10 15, 2015, sur https://code.google.com/p/oryx-editor/downloads/list

Oryx-Editor. (s.d.). *Oryx-Editor Database Architecture*. Consulté le 04 18, 2015, sur https://code.google.com/p/oryx-editor/wiki/DatabaseArchitecture

Patnaik, M. (2013, 05 08). *Big Data, Hadoop and Real BPM - The Connections*. Récupéré sur InforSysBlogs.com: http://www.infosysblogs.com/bpm-eai/2012/06/big_data_hadoop_and_real_bpm_-.html

*Python Data Analysis Library*. (s.d.). Consulté le 10 10, 2015, sur http://pandas.pydata.org/

Risi, W. (s.d.). Playing the Software Archeologist: Exploring and Conquering an Unknown Legacy System.

Riva, C., Selonen, P., Systa, T., & Xu, J. (2004). UML-based reverse engineering and model analysis approaches for software architecture maintenance. *Software Maintenance, 2004. Proceedings. 20th IEEE International Conference on*, (pp. 50--59).

Robles, G., Gonzalez-Barahona, J., & Herraiz, I. (2005). An empirical approach to Software Archaeology. *Proceedings of the International Conference on Software Maintenance (poster session)*.

Sadiq, J., & Waheed, T. (2011). Reverse engineering \& design recovery: An evaluation of design recovery techniques. *Computer Networks and Information Technology (ICCNIT), 2011 International Conference on*, (pp. 325--332).

Saliou, P., Ribaud, V., & others. (2007). Ingénierie du logiciel par immersion: allers-retours entre apprendre et faire. *3e colloque SRED'Constructivisme et éducation: Construction intra/intersubjective des connaissances et du connaissant'*.

Santosh. (2012, 05 21). *Use Hibernate or not?* Récupéré sur Stack Overflow: http://stackoverflow.com/questions/10682079/use-hibernate-or-not

Sawant, N., & Shah, H. (2013). Big Data Application Architecture. 9--28. Springer.

Schooff, P. (2013, 02 28). *BPM and big data: Exploring the relationship between two key technologies*. Récupéré sur ebizq.net: http://www.ebizq.net/topics/int_sbp/features/13447.html

Seemann, J., & von Gudenberg, J. (1998). Pattern-based design recovery of Java software. *ACM SIGSOFT Software Engineering Notes*, *23*, pp. 10--16.

Serge Demeyer, S. D. (2008). *Object-Oriented Reengineeing Patterns*. Square Bracket Associates, Switzerland.

Silver, B. (s.d.). BPMN 2.0 Handbook.

Sinha, V. (s.d.). BPM and Big Data - Why it Makes Sense.

Sinur, J., Schulte, W., Hill, J., & Jones, T. (2012). Magic quadrant for intelligent business process management suites.

Sjoberg, D., Yamashita, A., Anda, B., Mockus, A., & Dyba, T. (2013). Quantifying the effect of code smells on maintenance effort. *Software Engineering, IEEE Transactions on, 39*(8), 1144--1156.

Spener, M. (s.d.). *What is the easiest way to learn Big Data and where can simple step-by-step tutorials that can be done on a simple laptop be found*. (Quora.com) Consulté le 10 10, 2015, sur https://www.quora.com/What-is-the-easiest-way-to-learn-Big-Data-and-where-can-simple-step-by-step-tutorials-that-can-be-done-on-a-simple-laptop-be-found

Spinellis, D. (2003). *Code reading: the open source perspective.*

Storey, M.-A. (2006). Theories, tools and research methods in program comprehension: past, present and future. *Software Quality Journal, 14*(3), 187--208.

Storey, M.-A., Fracchia, F., & Müller, H. (1999). Cognitive design elements to support the construction of a mental model during software exploration. *Journal of Systems and Software, 44*(3), 171--185.

Storey, M.-A., Wong, K., & Muller, H. (1997). How do program understanding tools affect how programmers understand programs? *Reverse Engineering, 1997. Proceedings of the Fourth Working Conference on*, (pp. 12--21).

Stucchio, C. (2013, 09 16). *Don't use Hadoop - your data isn't that big*. Consulté le 12 02, 2014, sur https://www.chrisstucchio.com/blog/2013/hadoop_hatred.html

Tadonki, C. (2004). Universal Report: a generic reverse engineering tool. *Program Comprehension, 2004. Proceedings. 12th IEEE International Workshop on*, (pp. 266--267).

The Apache Software Foundation. (s.d.). *HBase on Cygwin*. (The Apache Software Foundation) Consulté le 10 15, 2015, sur https://hbase.apache.org/cygwin.html

The Apache Software Foundation. (s.d.). *MapReduce Tutorial*. (The Apache Software Foundation) Consulté le 10 15, 2015, sur

http://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

twitter.com. (2015). *plusieurs pages*. Consulté le 2015, sur http://www.twitter.com

Weisinger, D. (2013, 01 17). *Business Process Management: BPM Converges with Big Data and Cloud to Create Big Process*. Récupéré sur Formtek: http://www.formtek.com/blog/?p=3518

Wendy Newstetter, M. (1999). Reverse Engineering Or Design Recovery: Two Approaches To Uncovering Designing. *ASEE Annual Conference Proceedings, p 5071-5076, 2000, 2000 ASEE Annual Conference and Exposition: Engineering Education Beyond the Millenium*.

Weske, P. M. (s.d.). *Oryx Editor*. (Google Code) Consulté le 04 03, 2015, sur https://code.google.com/p/oryx-editor/

Wikipedia. (s.d.). *HBase*. Consulté le 10 15, 2015, sur https://fr.wikipedia.org/wiki/HBase

Wohlgemuth, G. (2010, 01 24). *how to handle 400 Billion rows in postgress*. Récupéré sur Coding and more: http://codingandmore.blogspot.ca/2010/01/how-to-handle-400-billion-rows-in.html

Wood, R. (2003). Assisted Software Comprehension. *Final report, June*.

Woodie, A. (2014, 01 27). *When to Hadoop, and when not to*. Récupéré sur Datanami: http://www.datanami.com/2014/01/27/when_to_hadoop_and_when_not_to/

Wu, X., Murray, A., Storey, M.-A., & Lintern, R. (2004). A reverse engineering approach to support software maintenance: Version control knowledge extraction. *Reverse Engineering, 2004. Proceedings. 11th Working Conference on*, (pp. 90--99).

Yan, Z., Reijers, H., & Dijkman, R. (2010). An evaluation of BPMN modeling tools. 121--128. Springer.

Yu, Y., Wang, Y., Mylopoulos, J., Liaskos, S., Lapouchnian, A., & Sampaio do Prado Leite, J. (2005). Reverse engineering goal models from legacy code. *Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on*, (pp. 363--372).