Internship Report

Master in Computer Engineering – Mobile Computing

Restructuring the Internals of Kettle/Pentaho Platform

Viswesvar Sekar

Leiria, 29th September 2017

This page was intentionally left blank

Internship Report

Master in Computer Engineering – Mobile Computing

Restructuring the Internals of Kettle/Pentaho Platform

Viswesvar Sekar

Internship Report developed under the supervision of Dr. José Vítor Martins Ramos, professor at the School of Technology and Management of the Polytechnic Institute of Leiria, co-supervision of Dr. Maria Beatriz Guerra da Piedade, professor at the School of Technology and Management of the Polytechnic Institute of Leiria and host institution supervision of Pedro Vale, vice president of engineering development at Pentaho.

Leiria, 29th September 2017

This page was intentionally left blank

Acknowledgement

My thanks first goes to my school, for providing me an opportunity to do my internship in a reputed company and allowing me to move from Leiria to Lisbon. I would like to thank my Professors (*Dr.José Vitor Martins Ramos and Dr. Maria Beatriz Guerra Piedade*) *for guiding me each and every step for developing my report and other things throughout my internship. Also my gratitude to my supervisor at Pentaho Pedro Vale, my Team Leader Peter and my teammates in 21-b, and personally my thanks to Duarte Teixeira, André* Jorge*, Steve Maring, Amin Khan, Gonçalo Neto, Duarte Leão, Hugo Figueira and all my colleagues at Pentaho for all the support that was provided throughout 9 months and to complete my internship successfully.*

I also want to thank my parents, brother and friends for the support to do my masters.

My sincere thanks,
Viswesvar Sekar

This page was intentionally left blank

Resumo

Face ao enorme crescimento de dados é necessário que as ferramentas de *Business Intelligence* sejam atualizadas e mantidas. A Pentaho é uma empresa de desenvolvimento de software na área do *Business Intelligence* que oferece integração de dados e *Business Analytics*, em que os seus produtos de software são mantidos e atualizados com frequência. Isto traz muitas vantagens na disseminação dos seus produtos de software. Este relatório apresenta o trabalho realizado no âmbito do desenvolvimento e manutenção do software Pentaho, bem como uma visão da sua arquitetura e processo de construção.

Palavras-chave: *big data*, integração de dados, *business analytics*, desenvolvimento de software, arquitetura, processo de construção de software.

This page was intentionally left blank

Abstract

Given the growth of data there is the need of business intelligence tools to be updated and maintained. Pentaho is a business intelligence company that offers Data Integration and Business Analytics, where the software is frequently updated and maintained. This takes a lot of advantages to widely spread the software tool. This report presents the work involved in the development of Pentaho software and its maintenance. Report also contains architectural view of Pentaho software and the build process.

Keywords: big data, data Integration, business analytics, software development, architecture, build process.

This page was intentionally left blank

List of Tables

This page was intentionally left blank

List of Figures

This page was intentionally left blank

List of Acronyms

AEL	**A**daptive **E**xecution **L**ayer
BA	**B**usiness **A**nalytics
BI	**B**usiness **I**ntelligence
CBF	**C**ommunity **B**uild **F**ramework
CCC	**C**ommunity **C**harting **C**omponents
CDA	**C**ommunity **D**ata **A**ccess
CDE	**C**ommunity **D**ashboard **E**ditor
CDF	**C**ommunity **D**ashboard **F**ramework
CDG	**C**ommunity **D**ata **G**enerator
CDV	**C**ommunity **D**ata **V**alidation
CE	**C**ommunity **E**dition
CI	**C**ontinuous **I**ntegration
CST	**C**ommunity **S**tartup **T**aps
DET	**D**ata **E**xploration **T**ool
DI	**D**ata **I**ntegration
DSDM	**D**ynamic **S**ystem **D**evelopment **M**ethod
EE	**E**nterprise **E**dition
ETL	**E**xtract **T**ransform and **L**oad
GUI	**G**raphical **U**ser **I**nterface
HDFS	**H**adoop **D**istributed **F**ile **S**ystem
IDE	**I**ntegrated **D**evelopment **E**nvironment
J2EE	**J**ava **2** **E**nterprise **E**dition
KETTLE	**E**xtraction, **T**ransformation, **T**ransportation and **L**oading of data
MVN	**M**aven
N – Tier	Multi - tier
OLAP	**O**n**L**ine **A**nalytical **P**rocessing
PAA	**P**entaho **A**nalysis **A**nalyzer
PAD	**P**entaho **A**ggregate **D**esigner

PDD	**P**entaho **D**ashboard **D**esigner
PDI	**P**entaho **D**ata **I**ntegration
PDM	**P**entaho **D**ata **M**ining
PDS	**P**entaho **D**esign **S**tudio
PIR	**P**entaho **I**nteractive **R**eporting
PME	**P**entaho **M**etadata **E**ditor
POM	**P**roject **O**bject **M**odel
PR	**P**ull **R**equest
PRD	**P**entaho **R**eport **D**esigner
PSW	**P**entaho **S**chema **W**orkbench
QA	**Q**uality **A**ssurance
QAT	**Q**uality **A**ssurance **T**eam
RDD	**R**esilient **D**istributed **D**atasets
VCS	**V**ersion **C**ontrol **S**ystem
XP	**E**xtreme **P**rogramming

Table of Contents

Chapter 1- Introduction

Nowadays, Business Intelligence (BI[1]) tools have become a basic requirement for most of the companies. It is no doubt that it will also have the long-lasting future and mandatory requirement to all modern business landscape. There are lots of advantages in using BI tools we will see about this in later chapters. Pentaho is one of the top BI software company that offers Pentaho Business Analytics, a suite of open source products, OnLine Analytical Processing (OLAP) services, reporting, dashboarding, data mining and Extract Transform and Load (ETL) capabilities [1]. This document will detail about the developed work and acquired knowledge during the internship period, conducted for the course masters in Computer Engineering – Mobile computing, Polytechnic Institute of Leiria, during the academic year 2016/2017. This internship took place at the company Pentaho, Oeiras Lisbon from December 2016 to September 2017.

1.1 Internship Goals

The objective of the internship was to be a part of the development team and to redesign Pentaho platform, restructuring the internal organization of the Pentaho code base. Pentaho creates and develops a big data analysis platform. It comprises both data integration perspective (graphical design of ETL process) as well as their execution both locally or in the server mode (distributed execution is also possible) and the data analytics module – the capability of analyzing and visualizing the data in different formats. The platform has been in active development for more than 10 years and as an open source tool that had multiple contributions over many years. As such solution architecture was recently reviewed to guarantee that the basic principle of the software architecture is kept, as proper modularization, separation of concerns. So, the sprint was actively working to address this redesign and conformance of the whole Pentaho code base to the latest standards. The goal of internship includes to actively participate in the definition and execution of this revision. So, items as conversion of the project to more modern build infrastructure of the proper segmentation of the application modules, developing the new interfaces and making such the application is in a working state after the changes. It will also be a part of Pentaho Engineering team which is in Portugal and United States. So, it is extremely likely to be in touch with several offices.

1.2 Motivation

As mentioned earlier BI is a groom for this century. The most important benefits of BI are, it removes guesswork, BI gives you a quicker response to your business-related queries and speedy answers. It helps to gain insights into the customer behaviour and much more. We are in the era of technology "democratized". It will sooner or later become BI democratized to all tech industries (small scale to large scale). The reason I choose internship instead of dissertation and project is that I wanted to benefit from the experience. I want to have a new challenge to learn, improve and develop new skills set of software in real time environments.

[1] *BI – Business Intelligence is a broad category of computer software solutions that enables a company or organization to gain insight into its critical operations through reporting applications and analysis tools.*

During my internship two major competencies are centralised, working as a junior software engineer, to work in an effective way to the organisation, which also leads to gain knowledge without affecting the main goals. As a developer to be part of a team, work actively with a team and contribute to the tasks. During 9 months of internship with Pentaho I felt encouraged.

1.3 Report structure

This document is structured in the following way — Chapter 2 (background) describes Pentaho structure and my role in the team, list of tasks assigned during internship, we will also come across other BI tools that are available in market, the next chapter is chapter 3, which is about the component architecture of Pentaho Data Integration (PDI). In chapter 4 it is all about software development methodologies which was adopted by Pentaho. Chapter 5 consists of development of Pentaho projects from local machine to the end server. Chapter 6, is conclusion.

Chapter 2 - Background

This chapter describes Pentaho, its internal structure in the business product development and my role in that structure. It also describes the task carried out during the development of each internship goal. About the team which I worked with and its process and finally literature review.

2.1 About Pentaho

Pentaho is a BI software company that offers business analytics a suite of open source products which allows data integration, OLAP[2] services, reporting dashboards, data mining and ETL capabilities. Pentaho was founded in the year 2004 by Richard Daley, its headquarters is in Orlando, Florida, USA and later Pentaho was acquired by Hitachi Data Systems in 2015. Now Pentaho is officially named as Hitachi Vantara. There is big competition in the market but still, Pentaho holds the top position according to magazine ADT mag [15]. This success is not only because they have strong clients but also the management with in the company. As an intern, I have experienced that Pentaho has a strong and friendly leadership, which makes the employee work with zest. After being a part of Pentaho, my interest leads me to know more about it. So, once I crossed across this which I would like to share here in this document. From Matt Casters, Chief Data Integration. I hope it has some importance to share here, to know the real success.

Short motivation
"Every second of every day, our senses bring in way too much data than we can possibly process in our brains." – Peter Diamandis, Chairman/CEO, X-Prize Foundation [17]

"You can have data without information, but you cannot have information without data." – Daniel Keys Moran*, an American computer programmer and science fiction writer.* [17]

As said earlier 21st is an era of business intelligence – big data. Day by day there are plenty of new tools, new feature is added to BI. Even though there are plenty of tools available the needs still exist and this pays a way for new origination to emerge and existing organisation to involve in developing. This is great success story that I have read so far, Appendix VI. Let the world knows it.

Pentaho provides:

- Server applications;
- Client/desktop applications;
- Open source Pentaho server plug-ins.

Pentaho has two types of license

- Pentaho Community Edition (CE) – Apache License version 2.0 (open source);
- Pentaho Enterprise Edition (EE) – Commercial License (paid version).

[2] *OLAP - Online Analytical Processing OLAP performs multidimensional analysis of business data and provides the capability for complex calculations, trend analysis, and sophisticated data modelling.*

These are the three products that Pentaho provides, Table 1. Server Application is a piece of software that host the content in server through plugins or publishing the files from the desktop application. The main features are displaying the dashboards, providing the report, OLAP analysis, etc. Table 2. Desktop Application is a core software Data Integration Engine and GUI apps, which helps users to run jobs and transformation. It supports all platforms (Windows, Mac, Linux). Table 3. Community driven tools and plugins which are on Pentaho BA Server as a stack which allow user to enable customization to BI Pentaho Platform.

Server Application

Products	Offerings	Latest version(EE)	Latest Version (CE)
Pentaho Platform Business Analytics (BA)	EE, CE	7.1	7.1
Pentaho Analysis Service	EE, CE	3.7.0	3.6.1
Pentaho Dashboard Designer (PDD)	EE	5.0.6	-
Pentaho Analysis Analyzer (PAA)	EE	5.0.6	-
Pentaho Interactive Reporting (PIR)	EE	5.0.6	-
Pentaho Data Access wizard	EE, CE	-	-
Mobile Device (app)	EE	5.0.6	-

Table 1 Pentaho server application products [1].

Desktop/ Client Application

Products	Offerings	Latest version(EE)	Latest Version (CE)
Pentaho Data Integration (PDI)	EE, CE	7.1	7.1
Pentaho for big data	EE, CE	-	-
Pentaho Report Designer (PRD)	EE, CE	3.9.1	3.9.1
Pentaho Data Mining (PDM)	EE, CE	-	-
Pentaho Metadata Editor (PME)	EE, CE	7.1	7.1
Pentaho Aggregate Designer (PAD)	EE, CE	7.1	7.1
Pentaho Schema Workbench (PSW)	EE, CE	3.5.0	3.5.0
Pentaho Design Studio (PDS)	EE, CE	4.0	4.0

Table 2 Desktop/Client Application products [1].

Community driven, Pentaho plug-ins

Products	Offerings	Latest version(EE)	Latest Version (CE)
Ctools	EE, CE	Various	Various
Community charting components (CCC)	EE, CE	Various	Various
Community Build Framework (CBF)	EE, CE	3.7	3.7
Community Data Access (CDA)	EE, CE	-	-
Community Data Browser (CDB)	EE, CE	-	-
Community Distributed Cache (CDC)	EE, CE	-	-
Community Data Generator (CDG)	EE, CE	-	-
Community Data Validation (CDV)	EE, CE	-	-
Community Graphic Generator (CGG)	EE, CE	-	-
Community Dashboard editor (CDE)	EE, CE	-	-
Community Dashboard Framework (CDF)	EE, CE	4.8	4.8
Community Startup Taps (CST)	EE, CE	1.0	1.0
Saiku	EE, CE	2.4	2.4

Table 3 Community Driven and Pentaho plugins [1].

2.2 Tasks

The object of the internship was to restructure the internals of Pentaho platform code base. Pentaho releases their updates or product every year. For the development, process Pentaho have used an adaptation of Scrum agile software development methodology, an internal policy of the company. This method is flexible, iterative and has an incremental approach to optimize predictability and control risk [2]. In agile scrum methodology the span of time is named as sprints. Within each sprint, the development team builds and tests a functional part of the product until the product owner accepts it and the functionality becomes a potentially shippable product. When one sprint finishes, another sprint starts. Scrum teams deliver product features in increments at the end of each sprint [10]. A product release occurs at the end of a sprint or after several sprints [10]. We will see more about this in Chapter 3. The list of each task done so far is attached in Appendix III. Next section is related to the planning phase

2.2.1 Planning

This section will describe the schedule for each project. It will also describe each phase association with different Pentaho projects.

The first three months of my internship was the period of training i.e. on-boarding which I had some training sessions on PDI and BA. And after a month during middle of December and January I was working with projects to acquire knowledge on Pentaho projects. Later the first week of February I was invited to my team (21-B) and started working on project with the team. I was also working on my report parallel to my work.

In concern to my internship plan the following table presents the schedule.

Tasks	Dec	Jan	Feb	Mar	April	May	June	July	Aug	Sep
On-Boarding/Training										
Restructuring the internals of Pentaho										
Mavenization										
Product Release										
Report writing										

Table 4 Internship Workplan.

2.3 About my team

My team is one of the development team in Pentaho engineering. The team is named as 2-1b, which was named after the Star Wars movie series. There are also other teams which hold their names based on the characters in Star Wars. There are total of 11 sprint teams. Specific details scrum of 2-1b team we will be described in the next chapter

2.3.1 Purpose

The 2-1b team will be focused on any items that significantly and positively impact the health of the Pentaho code-base (this reason leads to call 2-1b as code health team.) and reduce the difficulty of development by the Pentaho Engineering teams and open source community. This includes but is not limited to easing the development process by reducing project/code complexity, simplifying the ability to add valuable and correct automated testing (unit, integration, Quality Assurance (QA) automation), faster user and system build processes, reduction of “different” processes based on project and type of user (developer vs automated). This team may also be requested to handle release-oriented tasks which fall more into the “library/tool upgrade” and “security vulnerability” areas

2.3.2 Process

The team will function as a normal sprint team and will be composed of developers, QA[3] engineers from both manual and automation backgrounds, build engineer(s), and a Product Owner.

Members of the User Experience (UX) and Documentation teams will be included on an "as needed" basis since much of this team's work will not cause changes visible to the end user which would require UX or Documentation updates

This team will be expected to define and document any work processes undertaken and communicate that information to the rest of the engineering group for educational purposes. At that point, the other teams may be requested to engage on portions of the work based on their current work assignments and code/processes impacted. This work will become part of that team's sprint commitment and must not significantly impact the overall time-lines of the release commitments.

The Product Owner will confer with engineering stakeholders to form the team's backlog and define priority within that list. Once completed, the Product Owner will then work with the team to decompose the request list into actionable items and refine the priority of execution. Any item that does not have enough definition will not be executed upon until that definition is provided by the Product Owner.

The stakeholders for this team will be the engineering leadership team and (by extension) the developers, QA engineers, and build engineers within the engineering organization [15].

The developers on this team will be of differing levels of experience. In addition, any new developer that joins the engineering team will be added to this team temporarily until they gain experience with the Pentaho code base and are assigned to another sprint team [15].

As with any other sprint team, all members of this team will be assigned to this team for one release cycle. At the end of each release, cycle members may move/be assigned to other sprints team based on upcoming work assignments and the desires of the team members.

2.4 BI Tools Review

In this chapter, we are going to discuss about some of the similar BI tools available in the market. Advantages of BI which turns data in user information. There are actually plenty of BI Tools in today's market but for now let's discuss about MicroStrategy, Tableau BI and their products.

2.4.1 MicroStrategy

MicroStrategy is a BI software, providing integrated analytics, featuring mobile apps, and showcasing enterprise-grade security [5]. Its business intelligence platform allows leading organizations users to check the huge amounts of data that are stored across their companies.

[3] QA - Quality assurance is a way of preventing mistakes or defects in manufactured products and avoiding problems.

MicroStrategy involves in analytics, superior data, user scalability, etc. [5]. The MicroStrategy platform provides actionable information to business clients over the Internet and mobile devices, this mobile platform enables companies and organizations to build, deploy, as well as maintain mobile apps across a wide range of solutions by embedding transactions, multimedia, and intelligence, into apps [5]. An advantage of MicroStrategy, they provide mobiles application both in android and iOS, so the user can interact easily with any platform. For instance, from mobile to desktop application.

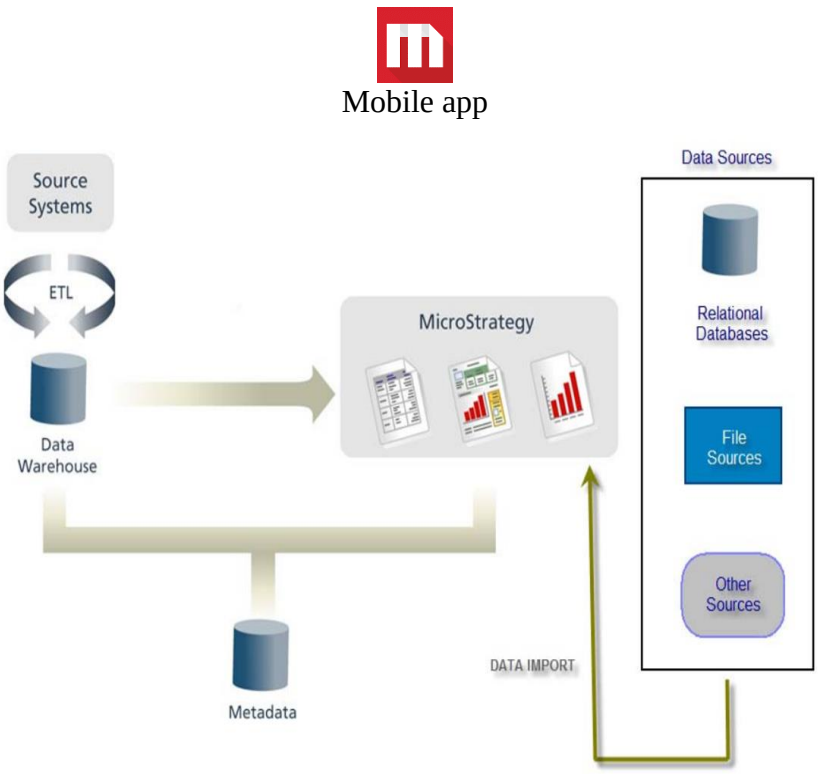

Figure 1 MicroStrategy component architecture [4].

MicroStrategy offers Real-time WYSIWYG[4] report design, dashboards, scorecards mobile app, big data solutions etc.

2.4.2 Tableau

Tableau is Software Company headquartered in Seattle, Washington. It provides a BI tool which includes data visualization products. Tableau current version stand of 10.3.1 with Security Bulletins updates. Tableau also provides cloud based services with scalable SaaS. Tableau Products

- Tableau Desktop (Explore visualize and analyze the data);
- Tableau Server (This is Web based application platform with meta data management);
- Tableau Reader (This is an open source product, viewing application which allows user to interact with package workbook created by tableau desktop).

Tableau has scalable, n-tier5 client server architecture that serves mobile, web and desktop applications.

[4] *WYSIWYG refers to software that accurately represents the final output during the development phase.*
[5] *n-tier stands for multi-tier architecture.*

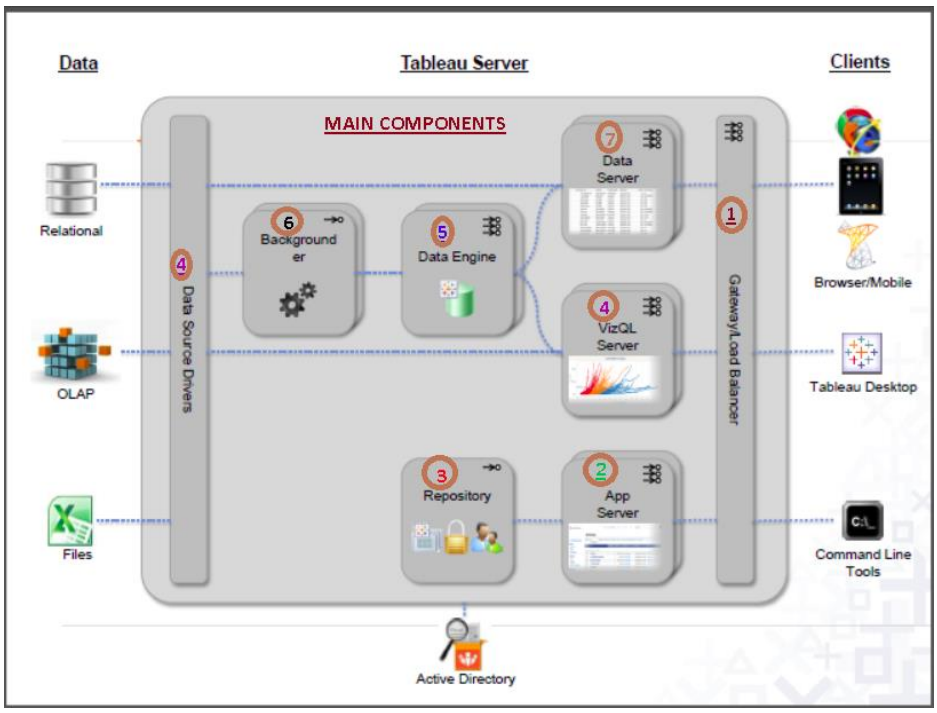

Figure 2 Tableau Component Architecture [13].

Comparison of Pentaho, MicroStrategy and Tableau

Comparison	Pentaho	MicroStrategy	Tableau
Features	Data integration Business Analytics Big Data Analytics Embedded Analytics Cloud Analytics Ad Hoc Analysis OLAP Predictive Analysis User-Friendly Interface Ad Hoc Reporting Customizable Features Performance Measurements Intuitive dashboards	Self-service analytics Big data solutions Software as a service (SaaS) Real-time WYSIWYG report design Scorecards and dash-boards Agile analytics Enterprise reporting Mobile	Toggle view and drag-and-drop List of native data connectors Highlight and filter data Share dashboards Embed dashboards within Mobile-ready dashboards Tableau Reader for data viewing Dashboard commenting Create “no-code” data queries
Available Platforms	MAC, Windows, Linux.	Mac, Windows, Linux, Android/IOS.	Mac, Windows, Linux, Android/IOS (web-based)
Types of Clients	Small Business | Large Enterprises | Medium Business	Large Enterprises | Medium Business	Large Enterprises | Medium Business
Offers	Open source and Enterprise Version	Paid version only	Paid version only

Table 5 Comparison of Pentaho vs MicroStrategy vs Tableau.

Chapter 3 - Pentaho Architecture

In this chapter, we will discuss about the component architecture and its internals. Pentaho updates are released every half-yearly. This chapter describes component architecture of version 7.x.

3.1 Component Architecture

Pentaho component architecture is divided into two components, integrated with a server called Pentaho Server.

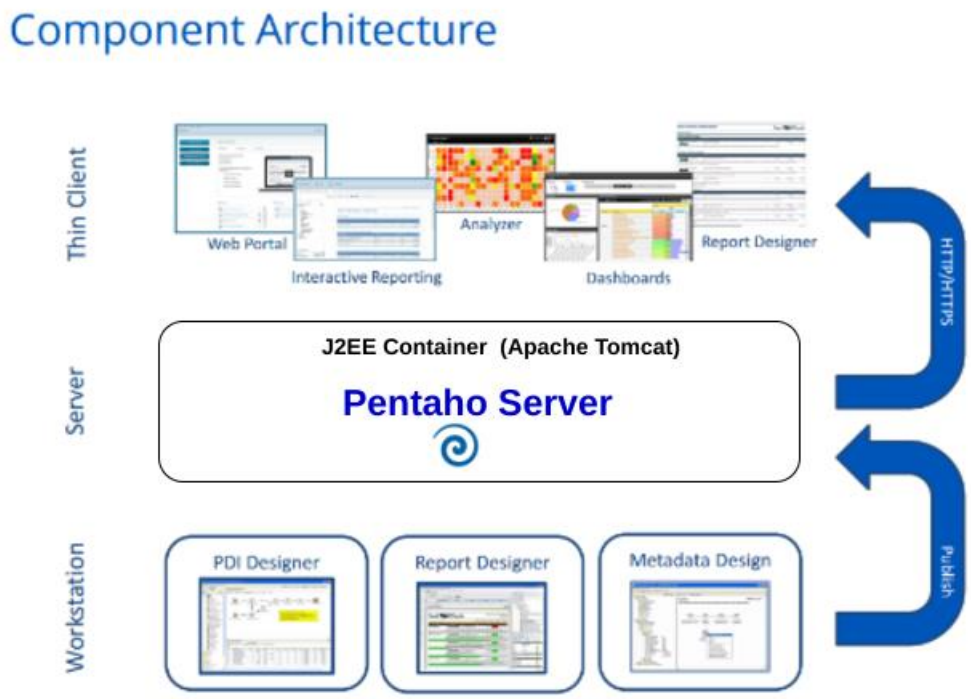

Figure 3 Component Architecture PDI 7.x.

Short Overview of Component architecture:

This component architecture is classified by three processes, Figure 3.

- Local Workstation
- Pentaho Server
- Thin client

Local work station is the initial process of where the user loads the data, it is also called a spoon. PDI supports mac, Linus, Linux operation systems. Spoon is the design interface for building ETL jobs and transformations, and resides on your desktop, Appendix I. Spoon provides a drag and drop interface allowing user to describe graphically.

Next the server component contain BA (Business Analysis server and Data Integration Server) transformations which can then be executed locally within Spoon, on a dedicated Data Integration Server, or a cluster of servers.

3.2 Internal Pentaho architecture

Internal architecture of Pentaho is a layered architecture. Each layer acts as a trans[6] of other layer. Top layer is data layer, middle level layer is server layer and client layer is the lower level layer.

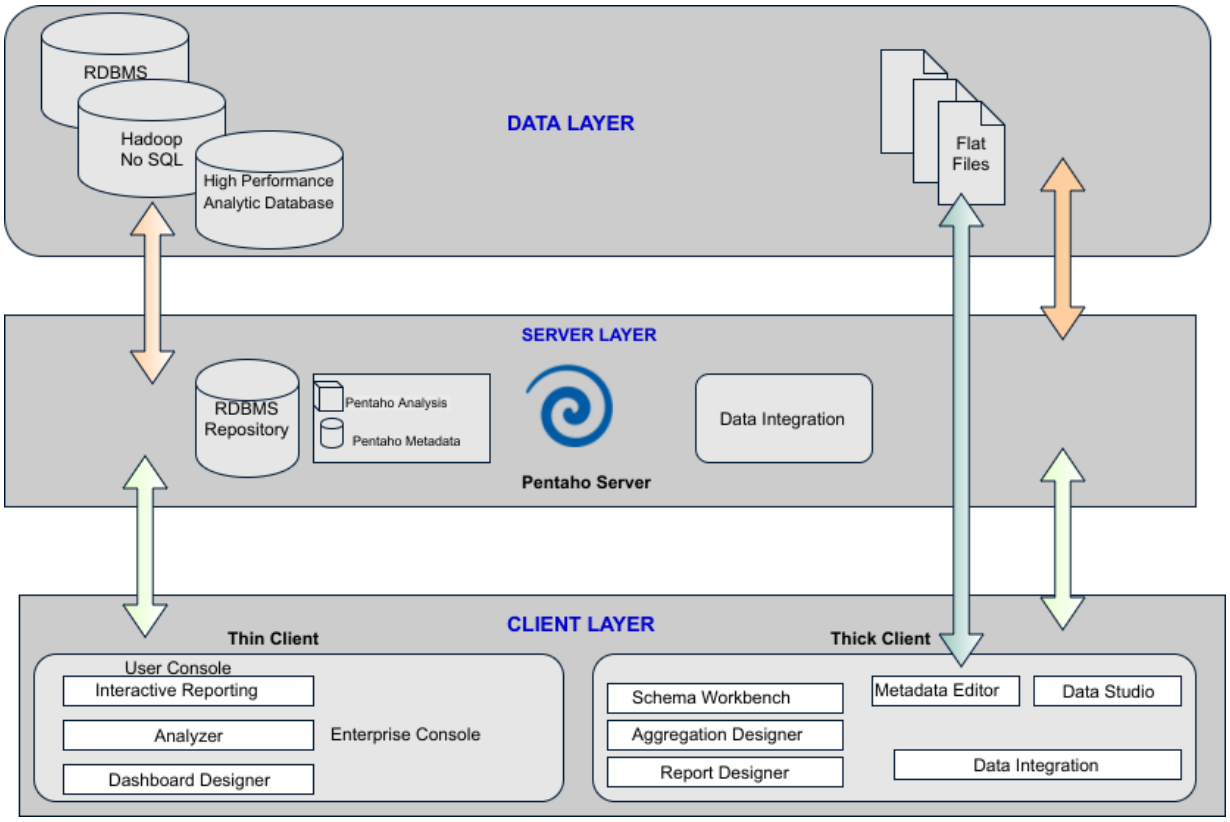

Figure 4 Pentaho Internal Architecture.

The above Pentaho internal architecture consists of three layers. Client layer is the low-level layer, the client layer is spitted into two thin clients and thick client. The thin client which runs on the server. Such as Community Dashboard Designer, Analyzer and Interactive Reporting. The thick client runs as a standalone which includes Schema Workbench, Data Integration (DI), Report Designer and Meta data Editor for mapping with physical database, Flat-files and business logic.

The middle layer is the server layer. Server layer consists of RDBMS repository and Data Models, such as Pentaho Analysis and Pentaho Metadata. The transformation that is executed in spoon with metadata [Appendix IV] this is where it handles trans meta data. This is used for deploying the report

[6] Trans refer to the layer/components which allows other layer/component to interact freely.

and make it available for the end user. It is also used for mapping. Data integration runs all jobs and transformation.

Data Layer is the top layer of the architecture which used to connect any data source. Additionally, Pentaho provides an excellent feature in the client Layer “C-Tools”.

C-Tools

The Pentaho platform offers users to use external components and plugins. The purpose is for extending standard functions. These plugins can be installed on Pentaho platform which enable users to do customization to the BI platform.

Dashboards	Admin	Development	Data
Community Dashboard Editor (CDE)	Community Distributed Cache (CDC)	Community Build Framework (CBF)	Community Data Access (CDA)
Community Dashboard Framework (CDF)	Community Data Validation (CDV)	Community Text Editor (CTE)	Community Data Browser (CDB)
Community Chart Components (CCC)	Community File Repository (CFR)		Community Data Generator (CDG)
Community Graphics Generator (CGC)	Community Startup Tabs (CST)		

Table 6 CT-Tools.

3.3 New features in Version 7.1 PDI

This section describes the view of features in Pentaho 7.1, a brief description of capability and how it is used, some importance of those features and some general knowledge about spark in-order to get prior knowledge for the readers.

3.3.1 Introduction

The Pentaho 7.1 is a set to be big release to enhance user experience, this feature is mainly focused on new Data Explorer capabilities such as Data Inspection, BI prototyping, etc. Another important feature is Big Data Processing this implementation has a cool feature “Adaptive Execution Layer (AEL)”, which was implemented for flexible data processing with different execution engines. My team 21-b have contributed to this feature, by making documentation, handling maven related part such as AEL assemblies. For sprint related work refer to Appendix V.

Before heading to core concept of AEL let us get to know some basic functionally about Apache Spark, spark abstraction and concepts.

3.3.1.1 Apache spark

Apache spark is a general-purpose computing engine offered by apache, which will allow to run batch interaction and streaming jobs on the cluster using same unified frame. Before heading to core concepts of Pentaho adaptation on AEL we can have basic understanding about Apache spark.

3.3.1.2 Spark abstraction and concepts

Spark concepts involves some main components as listed below. The reason why apache spark is widely used now than apache Hadoop because of this component. Spark engine are 4 time faster than map reduce.

- RDD: Resilient Distributed Datasets
- DAG: Direct Acyclic Graph
- Spark content
- Transformation
- Actions

Apache Spark is a flexible data transformation engine which can perform work across a cluster of machines. Central to this execution is the concept of a Resilient Distributed Dataset (RDD). RDDs can be constructed from an existing set of data such as flat files on disk or data in-memory. In this case the data is split up and distributed across the cluster where execution against the data can be done in parallel.

RDDs may also be created by leveraging data already existing on the cluster nodes in the form of various Hadoop resources, HDFS, HBase, etc. The advantage being that this data is already parallelized in the cluster and only the operations on that data need to be sent across cluster.

3.3.1.3 Data Explorer Tool

The Data Exploration Tool (DET) for Pentaho is a tool that allows users to inspect the data flowing within transformations in PDI or data sources on the Pentaho Server.

3.3.2 AEL Spark on Pentaho

The following section describes AEL on Pentaho before and after implementation and advantages etc.

3.3.2.1 Before Implementation

PDI lacks a clean API for execution of transformations and jobs (Appendix II). User must code directly to the internals of the classic engine such as Trans.java. Pentaho Clients such as Carte and Spoon are also written directly to these classes. This is a major complicating factor for making large changes to the existing engine. To support future engines a clean API must be introduced for clients and the engine itself to use.

3.3.2.2 After Implementation

Adaptive PDI Execution Engine (AEL), the new engine is to enhance (future-proof) the current engine to allow PDI transformations and jobs to take full advantage of the emerging distributed computational environments such as Apache Spark, Flink and Beam.

3.3.2.3 Data Processing Using the Adaptive Execution Layer (AEL)

AEL are used at current version for running transformation in different engines. Now user can run the transformations using a spark execution engine. AEL builds a transformation definition for Spark, which moves execution directly to the cluster, leveraging Spark's ability to coordinate large amounts of data over multiple nodes [12].

3.3.2.4 Drill-down Deeper on Data in-flight

PDI can now connect to Microsoft Azure HDInsight cluster, The Azure HDInsight cluster is a Microsoft web service for big data processing and analysis that is an alternative to hosting in-house cluster computing. Pentaho supports both HDFS and WASB (Windows Azure Storage BLOB) storage [12].

3.3.2.5 Big Data Security

Pentaho Server support of Kerberos impersonation has been expanded to Hortonworks Hadoop clusters. This allows multiple PDI users to access Kerberos-enabled Hortonworks Hadoop clusters as multiple authenticated Hadoop users [12]. The Pentaho Server controls access to specific data within Hadoop – based on a user’s role in the organization – to enforce enterprise data authorization, Kerberos authentication, and secure impersonation on Hadoop clusters [12].

Chapter 4 – Software Development Methodology

This chapter describes the software development methodology adopted by Pentaho, as for the methodology, an adaptation of Scrum for each internship goal was used. Scrum is an agile methodology that uses a group of techniques based on iterative and incremental development [11].

4.1 Introduction

During the development phase, we have used an adaptation of Scrum, an agile methodology to create the business prototype and share interests. Jim Highsmith says that being agile means being able to "Deliver quickly. Change quickly. Change often." [10]. It consists of a group of techniques based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development and delivery, a time-boxed iterative approach, and encourages rapid and flexible response to change [11]. There are some alternative agile methods, such as Extreme Programming (XP) [12], Scrum [8] and Dynamic Systems Development Method (DSDM) [13], waterfall model etc.

For the development process, we have adopted an agile methodology based on some concepts of Scrum, as this is the methodology adopted by Pentaho. The used methodology does not follow a rigid specification of Scrum but adopts the basic concepts that define it. This allows a better adaptation to the company projects and a more flexible internship. In the next section, we will give a description of what Scrum consists, the terms used in it and its usage within Pentaho company projects.

4.2 Scrum Methodology

Scrum is an agile software development methodology. This method is flexible, iterative and has an incremental approach to optimize predictability and control risk. In Scrum there are some specific terms, such as product backlog, sprint, sprint backlog and standups, Figure 5 [9]. The product backlog is owned by the product owner, responsible for its availability, content and ordering. It is an ordered list of everything that could be needed in the product and is the single source of requirements [8]. Scrum is based on sprints which is a basic unit of development, with a fixed-length periods of time of one to four weeks (usually two at Pentaho-15 days) during which a list of things is performed or certain functionalities are delivered. The set of features that go into a sprint come from the product backlog. Which backlog items go into the sprint (sprint backlog) is determined during the sprint planning meeting. At the end of each sprint, a sprint shutdown meeting is held. During this meeting, the Scrum team shows what has been accomplished during the sprint and adapts the product backlog

if necessary. Based on that and any changes to the product backlog, attendees collaborate to decide what to do next [8].

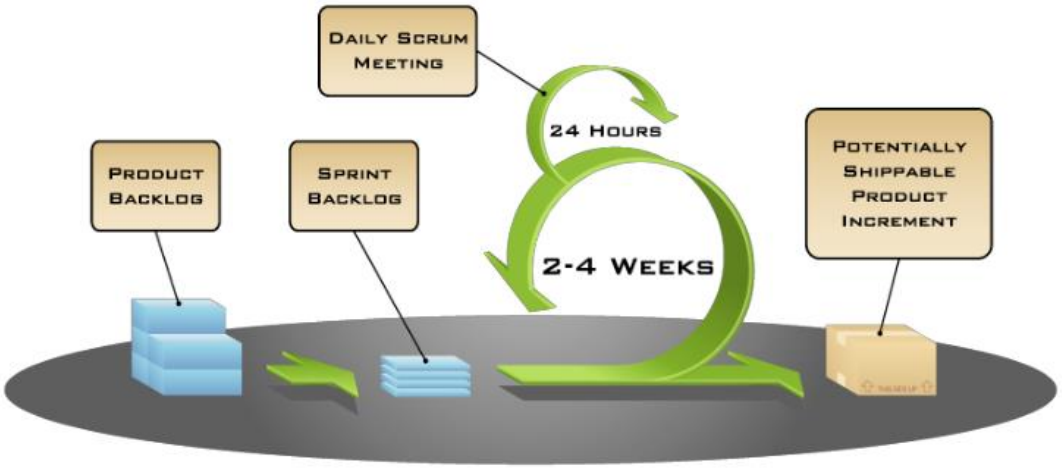

Figure 5 SCRUM diagram [9].

On Scrum, there are some core roles, which are committed to the project in the process and that are responsible for developing the product and representing the scrum team. The roles are:

- **The Project Owner**: Represents the stakeholders and is the voice of the customer or someone who has his vision. The product owner writes the user stories, prioritizes them, and adds them to the product backlog. He is accountable for ensuring that the team delivers value to the business [14].
- **The Scrum Master**: He is responsible for making sure that the team lives by the values and practices of Scrum and for removing impediments to the ability to achieve the sprint goal. The scrum master is not the team leader, but acts as a buffer between the team and any distracting influences, protecting the development team and keeping it focused on the tasks at hand [14].
- **Development Team**: It is responsible for delivering product increments at the end of each sprint. A development team is self-organizing, and it is composed of three to nine people who do the actual work (analyses, development, testing, documentation, etc.) [14].

4.2.1 Scrum on Pentaho

The scrum implemented into the engineering process is to have an incremental progress. JIRA Project management tool is used for this methodology. So, Pentaho Scrum is applied towards all engineering development team so far there is 11 team in Pentaho.

Now, let's see in concern about the 2-1b team, which was the team I worked with.

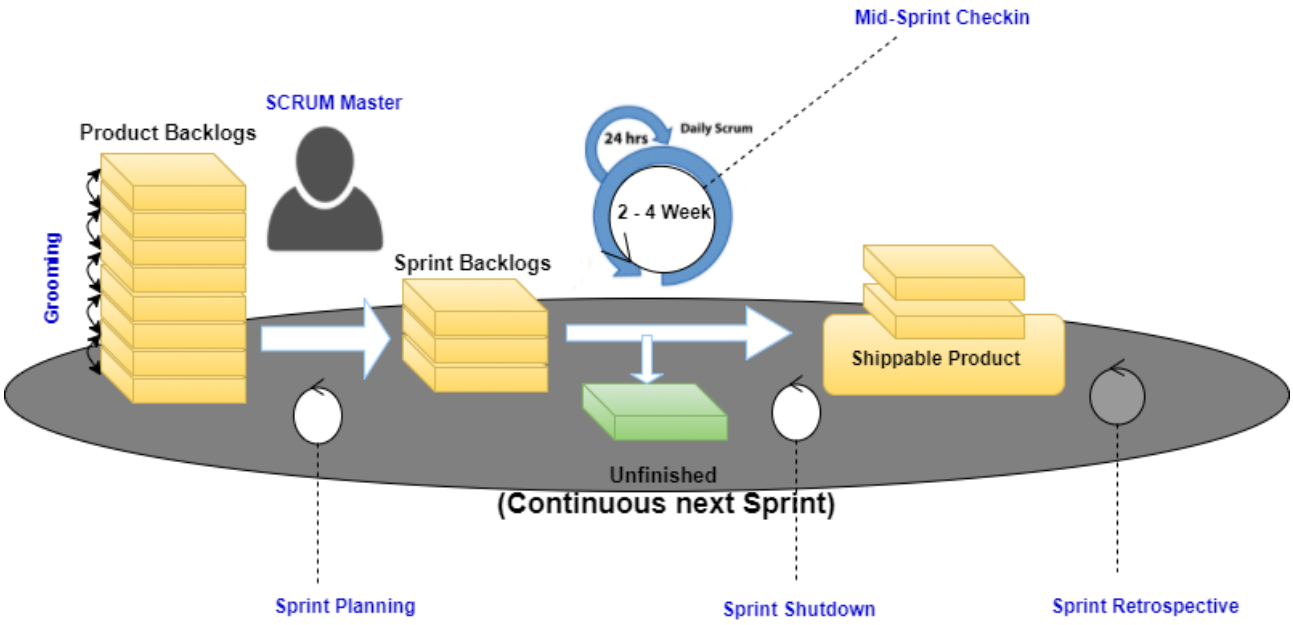

Figure 6 Scrum of 2-1B.

In 2-1b Team, the role of the product owner is taken by the Engineer Peter Minutillo, he also plays the role of Scrum Master. The development team is considered as one element. There is also a build team member, this is taken by Larry Grill. The sprint duration established 2 weeks around 14 days, where the whole team talks about the project status and established a new group of goals to the next sprint. During all the development time, the project was supervised by the scrum master, who has helped in some tasks and gave expert guidance. Below, Figure 7, are the scrum board view in Jira. Now let's see the things we do during the meeting since there are several meetings within the team during the sprint.

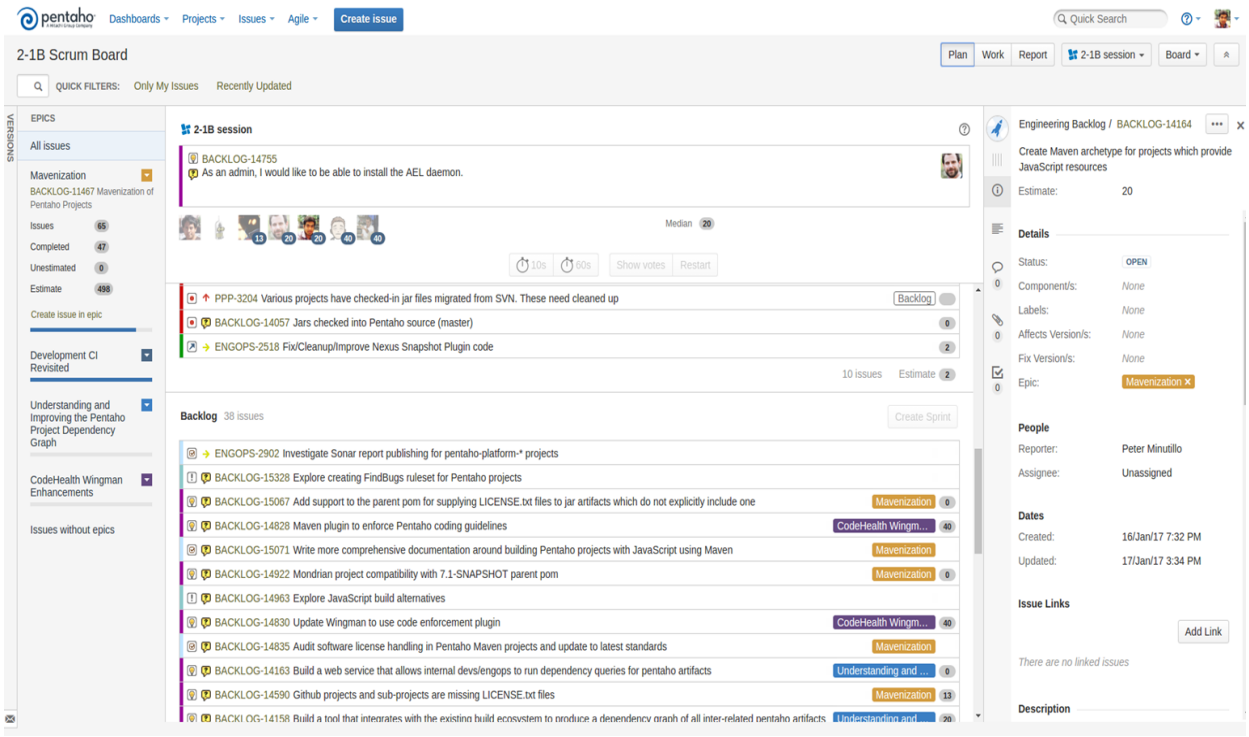

Figure 7 Scrum Dashboard (Jira).

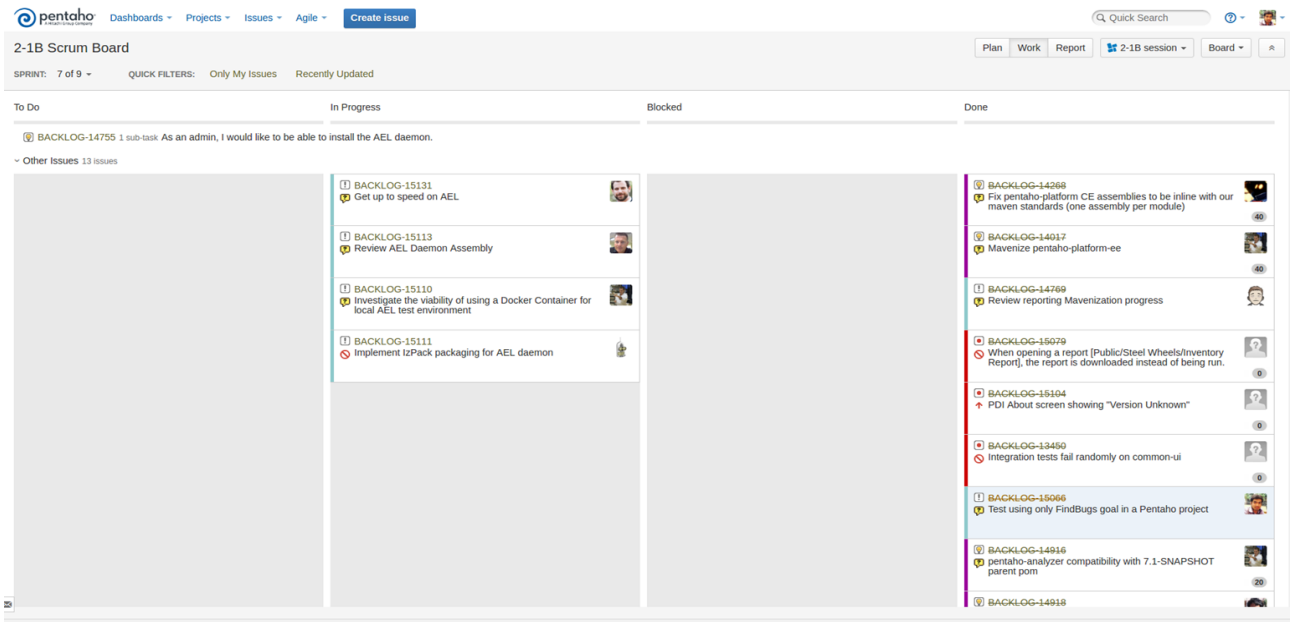

Figure 8 2-1B Sprint board (Jira).

4.2.1.1 Sprint Team Charter

The sprint team charter is a list of things that each team agrees is necessary to their success. Each team member commits to adhering to the charter. This sets a common expectation for and between each member of the team. Ideally, the charter would be created during sprint zero, but can be introduced and/or modified at any time, with the buy-in and agreement from each member of the team.

4.2.1.2 Team Standups

Stand ups are all about team coordination and commitment. It's a short, focused period of communication for the entire team. The output is a plan for what will be accomplished in the next 24 hours.

It’s easy to get off track and turn stand up into a bloated status meeting (this is the antithesis of a standup). Each team member's goal is to answer just three questions:

1. What did you do yesterday?
2. What will you do today?
3. Are there any impediments in your way?

4.2.1.3 Grooming the Backlog

Backlog grooming is a way to prepare the scrum backlog for the sprint planning meeting. This usually includes adding new stories to epics, extracting stories from existing epics, and estimating effort for existing stories. Stories are to be broken down into work that the team expects they could complete during a sprint. This often requires that the story defines a clear scope of work that may have known limitations and assumptions [15]. Limitations, assumptions, acceptance criteria and the definition of

done are to be clearly defined in the story. Stories should be groomed until the team agrees that they are ready for sprint planning.

Teams often groom during the planning session as well, which is more typical as described in traditional agile books. This is perfectly acceptable, and whether grooming is completed during the grooming session or during planning is a choice left to the team.

It is the responsibility of the product owner to clearly describe what is to be developed and why the team is building what they are building. Everyone on the sprint team must attend grooming

Each team has multiple times reserved on the shared sprint schedule to accommodate multiple grooming sessions during their sprint. This will allow the grooming meeting to be shorter and promotes multiple iterations for discussion and discovery.

4.2.1.4 Sprint Planning

In terms of planning, we have sprint planning meeting which lasts for an hour or two, which includes all the members in the team along with the build team take part. To discuss what to work in the current sprint and next sprint, the team will also decide the start date and end date of the sprint activities of a project.

In the purest implementation of Scrum, during the Sprint Planning, the Product Owner describes the highest priority backlog items to the team, which can be consider working first. Every member in the team asks questions until it is understood tasks and finally the team makes a commitment.

At Pentaho, we also have Grooming meeting (formally, these may be considered Estimation Meetings [15]) to understand and estimate upcoming work on the backlog. This helps the team to estimate the backlog, going into a Planning Meeting than could be accomplished during a single sprint. These Grooming Meetings greatly facilitate the Planning meeting to be more of an exercise in capacity planning and formalizing a commitment. The result is the same, we have groomed stories broken down into tasks with estimates added and committed to a sprint. The Product Owner is expected to be prepared to discuss about 2 sprints worth of stories.

Formally, the output (artifacts) of a Sprint Planning meeting are:

1. A Sprint goal (a short statement describing the sprint objective);
2. A Sprint backlog (items being committed to for the upcoming sprint).

Ask the team members what their availability will be during the upcoming sprint so that the commitment matches the capacity. Occasionally, a team member will be unavailable for some portion of a sprint due to vacation, training or other reasons. Identify these gaps so that the commitment of the team does not put unnecessary burden on the rest of the team. The goal is to commit to a reasonable and sustainable level of work to prevent "Sprint Exhaustion" (aka burnout).

4.2.1.5 Sprint Shutdown

The purpose of a sprint is to deliver working software that has been tested and documented. The sprint shutdown, more commonly known as the sprint review meeting, is held at the end of a sprint to showcase these deliverables and most importantly, to elicit feedback.

The review meeting should be kept informal. Discussion is therefore encouraged as the team collaborates with the product owner, customers and management to continue the feedback loop to adapt and refine the future direction of the backlog. It is not required to give a demo during a shutdown meeting. A demo may be used as a tool to facilitate the discussion. To highlight the informality of the review, it is suggested that the use of PowerPoint is forbidden. Although Pentaho often requires that demos use nightly builds, some suggest that a developer build is also acceptable [15]. This reinforces the importance of an informal review.

The review is not supposed to be a distraction for the team. The informality of the meeting suggests that we discuss the work that was completed without stressing about and overly preparing for the review. Some guidelines indicate that preparation over an hour is too much.

Seek feedback from everyone who may be present during the meeting. Ask for suggestions on how to deliver even better product in the future.

Although we have not typically done this, the Product Owner is supposed to be the presenter and facilitator of this meeting. At the very least, the Product Owner should weigh-in with comments about the backlog and velocity to give ballpark figures on the dates of completion for upcoming items on the team's backlog. The review should not be the first time the Product Owner is privy to the completed work, nor is it the time and place for the Product Owner to give formal sign-off.

Since it is a goal of Pentaho to have cross-functional Product Owners, it is strongly recommended that Product Owners attend shutdowns for other teams.

4.2.1.6 Sprint Retrospective

In this section, we'll cover what a retrospective is and what it isn't, and define best practices for making them more effective.

- What They Are?

At the core, a retrospective is simply a safe meeting to highlight what is working and what is not working. The purpose of a retrospective is to create an even more effective team.

- What They Aren't?

A retrospective is not the outlet for blame or negativity, nor is it the necessarily the forum to solve problems.

Best Practices

There are many pitfalls that teams will periodically fall into and there are plenty of helpful techniques to steer away from these situations. Since a retrospective is a meeting with the team, it is important to focus on effective use of everyone's time.

1. Encourage a safe environment to express opinions. This will be established over time with trust. When soliciting feedback, the facilitator should ask "What else?" not "Anything else?" This promotes the overall purpose of the meeting and begs for more information instead of assumes whatever had been said [15].

2. Be sure to get opinions from everyone, especially those who are typically introverted or shy about such discussions. Instead of only gathering opinions from those who shout it out, prompt others who often take a back seat [15].
3. Focus on positives, most negatives can be spun in a positive light. It's not about where we've been, it's about where we're going.
4. Keep track of previous notes, take time during the retrospective to revisit the notes and see how the team is doing [15].
5. If something went wrong during a sprint, this meeting is not about finger-pointing, it is about identifying areas of improvement.
6. Keep the format fresh. Occasionally, vary the technique/format of the retrospective. With new teammates, use ice breakers to get conversation started.
7. The facilitator needs to be in control and curb negativity.
8. Remain focused, the meeting should create action items for the team to take from the meeting, not solve in tangents during the meeting [15].
9. Don't stop doing retrospectives because you think things are going OK.
10. Spend some time before the meeting to prepare notes and ideas to share.
11. Hold the retrospective meeting immediately after the shutdown or as soon as possible thereafter [15].
12. Vary the role of the facilitator, the location and the format.
13. Just as a "Scrum of Scrums" is effective, so too is a "Retrospective of Retrospectives."

Below, is attached a sample calendar which occurred during the month of March, Figure 9.

Figure 9 calendar - Team Meeting

Chapter 5 - Development

In software development planning is the initial process in which product road map will be defined by the leaders at Pentaho, which includes updates to the next release and some improvements for the existing products. This chapter aims to report all the work involving the development of the software. We will see how each project has been assigned and processed. Also, we will come across how Pentaho builds the product and we will also compare the development work flow with 21-b team.

In earlier chapter, we describe the methodologies that Pentaho used to adapt for development (SCRUM). Now we will see the development process step by step.

5.1. Version control

Version control is a software tool used by software development teams to manage changes to source code over time, it keeps tracking of every modification made to the code. If a mistake is made, developers can revert and compare to the earlier version which helps them to fix it. Version control provides two primary data management capabilities.

- It allows user to lock the files so it can be edited;
- It allows user to track the changes.

There are many tools available for version control system, e.g., SVN, GIT, Mercuril, Bazaar, etc.

Git is the version control software used at Pentaho. There are several advantages by using git, such as Git provides a cloud collaboration so that the developer code can be viewed by other developer, the code can be reviewed and can also be reverted if needed. Git is also an open source software. The GitHub repository of Pentaho can be found at *https://github.com/pentaho.com.*

Since PDI is open source there are some repository available at Pentaho as open source, and some are private which can be accessed only by members within the Pentaho organization. Now let’s discuss the entire process in upcoming sections.

5.1.1 Stage and commit code

On the first step, fork down the repository which we would like to work with on GitHub and later clone the repository, clone is used to work with a repository locally (i.e. download it). Get the URL for the repository on your account by clicking the green "Clone or download" button while viewing your now-forked repository. Copy that URL and run the 'clone' command in the location on your machine where you want the files to be stored. All this process could either be done manually using the commands or by using desktop applications such as source tree, bitbucket or GitKraken based on the operating system. After working with the project, we should update the changes. The first step after the changes should be involved by staging it locally, as we discussed earlier the staging can be

either done by command line or by any application. The command for staging is as followed ($ git add.) for adding all changing to staged environment if the user prefer to add one by one ($ git add) followed by name to the module. After staging the user must commit the changes before push to the repository, committing is followed by the command line ($ git commit -m “any message”). After this process, the changes are staged and committed locally, so the user must push the changes to the remote repository. The command line followed to push ($ git push origin branch/master).

Here there is also another useful process called `Squashing´ which occurs before pushing the changes.

5.1.2 Squashing

Squashing is essentially the combining of multiple commits into one. It is important to save the work, so commit often. If the user makes multiple commits that are similar in work or feature, squashing would help for the sake of clarity. Squashing can be done as follows:

#	Action/Command	Details
1	git reset --soft <ID of commit before your first>	IT IS IMPERATIVE THAT USER USE THE "--soft" FLAG. A "--hard" reset will throw away all the changes and, while it can be done, it is difficult to recover them. user can determine the ID of the commit before the first one made during the most recent coding session by using the "git log" command. (the default flag is "--mixed" which is similar to "–soft", except that user will have to re-"git add" the files that he/she would wish to then commit.) a ref-log entry is made at every commit. These entries can be viewed via the "git ref-log" command. User can use this information in conjunction with another "git reset <target ID>", or other git command, to recover.
2	git commit -m <a string>	Notice that user do not need to use "git add" again, since the "--soft" flag will keep the changes queued up . 'commit -m' will commit these changes to the local repository. The string user enter will be tied to this commit. Provide a description of the changes made by the user.

The following graphic differentiates between different reset flags:

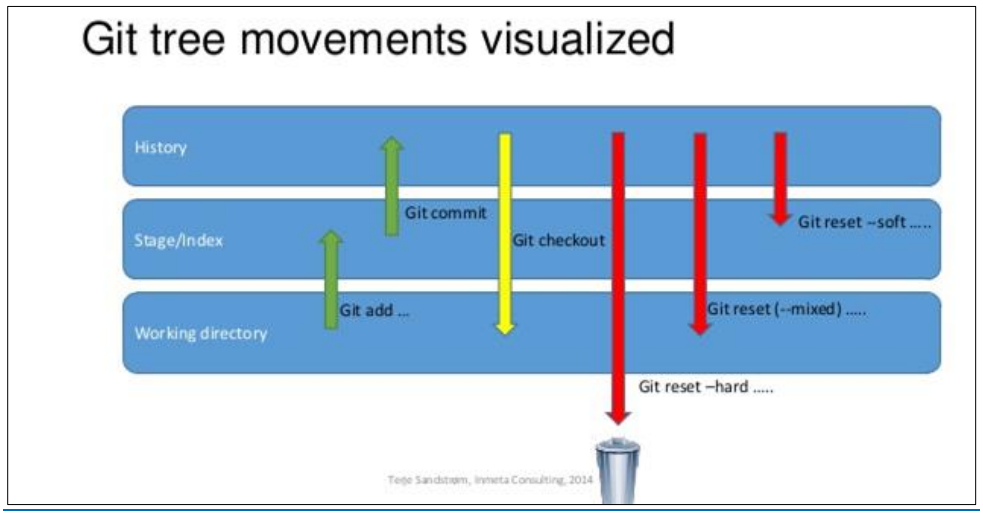

Figure 10 Git tree movements visualized [15].

5.1.3 Open Pull Request

After pushing the changes there should be a process called Pull Request (PR), which every person who pushed the changes to the repository should open a PR to merge it to the master branch. PR can easily be open by clicking the PR button on the Git web application. And there is also an option called comment where the person can tag any one or leave comment regarding the changes that are made. It would be an ideal option to tag the entire team. The figure below presents a sample of a PR.

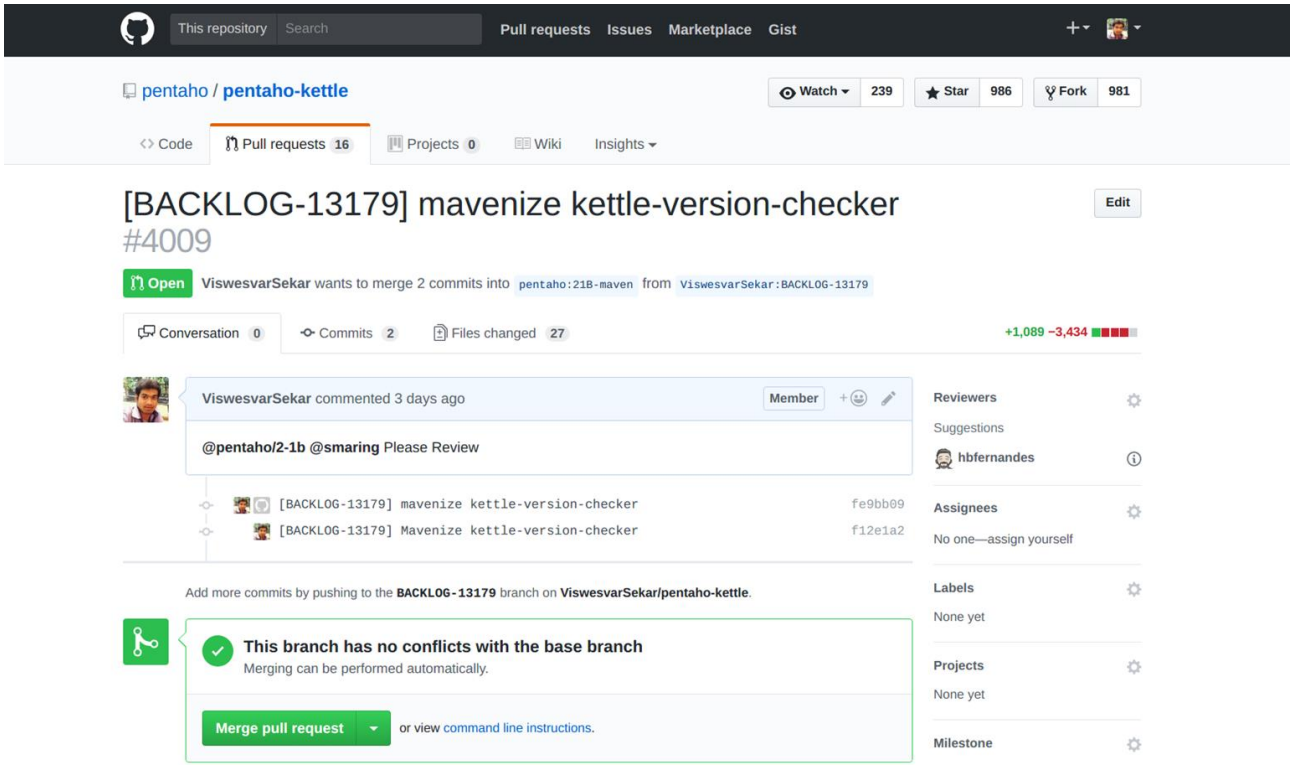

Figure 11 Sample view of an open PR.

This is for the time where the PR will be left open for the review, for the changes made. The peer review will be done by anyone who would like to be noted.

5.1.4 Wingman successful build

The Wingman (wingman.pentaho.com) is a utility system to validate code contributions before they are evaluated for their technical merit [11]. The general idea is to automate as much of the quality checks as possible upfront, thus leaving the developers to focus on the technical aspects of the code.

Wingman is entirely automated and does not need to be launched manually, it also regularly scans incoming pull requests and triggers the builds as required [11]. It is however possible to launch builds manually. To do so, one should trigger the wingman job, using proper parameter values. As a matter of fact, triggering the job manually adds to the queue and makes the builds slower for everyone.

The list of checks that Wingman performs:

Build - A pull request should not modify the default builds from Pentaho build team. The necessary override can be added to the project;

Coverage - The code must be tested appropriately. Minimum 80% for the new classes. No drop of coverage for the existing classes;

Files - A simple list of the files modified by this pull request;

Licenses - All files must have a proper license as their header;

Style - The code must adhere to the style guidelines;

Tests - The unit and integration tests must run successfully.

5.1.5 Merge by a peer

After Wingman Successful the PR would be merged. The merge will be with master branch this change can be revert if anything goes abnormal. We will see next process in upcoming sections.

5.2. Build Process

In previous section we described the build process so in this section we will have an architectural look of what happens in each phase of the build process, starting from the local work station to the release environment, Figure 12.

5.2.1 Explanation

Local and Wingman

Step 1: Local machine which involves certain operation, it is in need to do those operation based on the project. Here we see that the first phase local and the second GitHub. The Project are tested locally and then pushed to GitHub, a PR from the origin branch should be open (Step2). Step: 3 Wingman, as we have discussed on the previous section 1.4. Wingman[7] does validation at once a PR is open, after the Wingman validation the PR will be merged.

DevCI

The next phase is DevCI, this is the continuous integration build server where the builds are trigged from the developer's git commit. It creates the artifacts in the "http://build.pentaho.com", SNAPSHOT versions. This phase runs unit test. User can also trigger the build manually by using the artifact id/number. The ftp, http is published to internal consumers DMZ[8] (http://resource.pentaho.org).

[7] *Wingman does the checks for each commit on the PR.*

[8] In computer networks, a DMZ (demilitarized zone) is a physical or logical sub-network that separates an internal local area network (LAN) from other untrusted networks,

Release Infrastructure

Cerberus is the nightly build server which builds the product for QA testing. This also fetches from Github (PR) to build on the server, artifacts are created and hosted at http://build.pentaho.com/hosted directory, creates Quality Assurance Team (QAT) versions for QA testing. Cerberus runs unit test and integration test. Build are triggered once a day during night. Cerberus is managed by the Engineering Operation team.

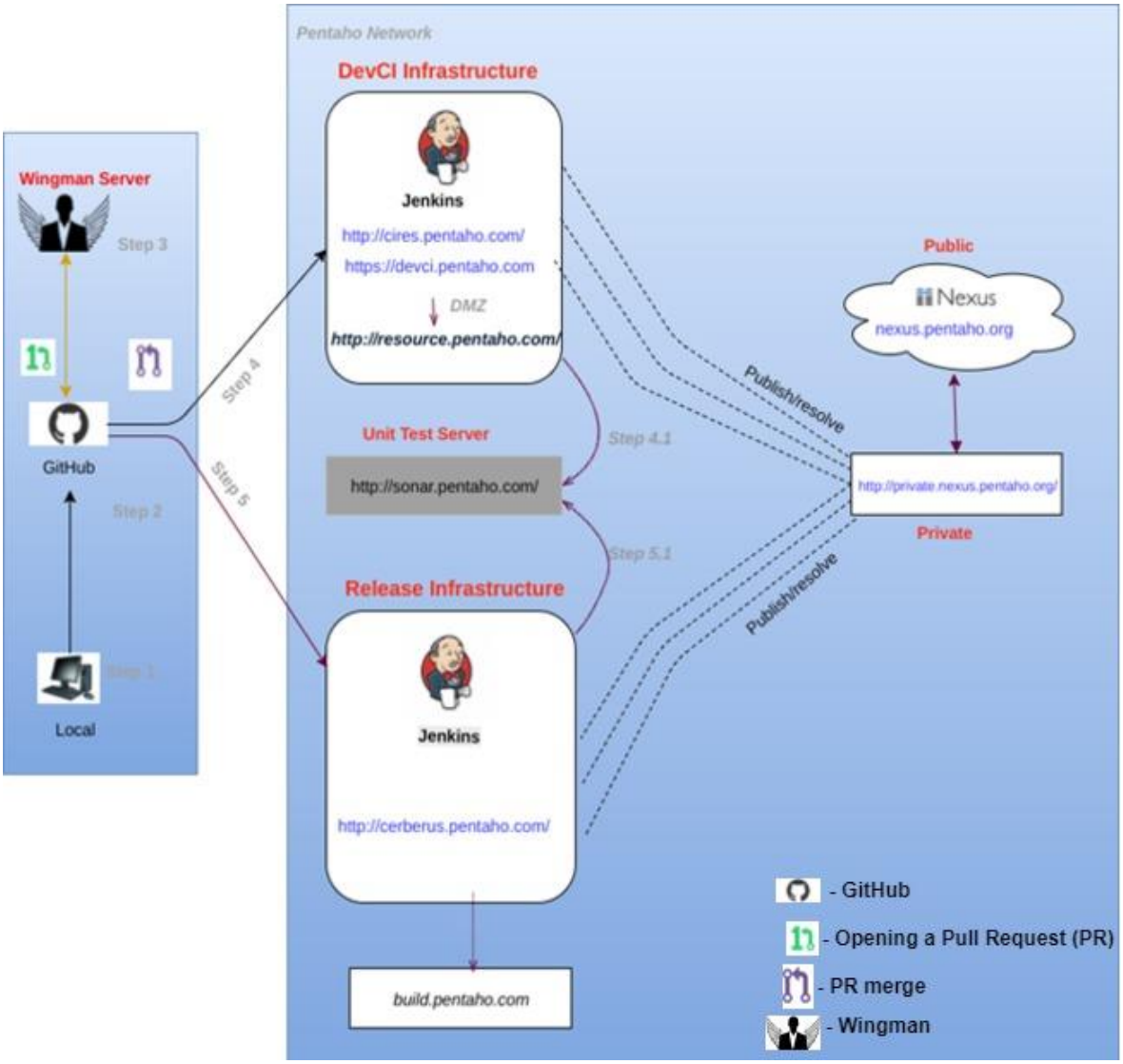

Figure 12 Architecture of Pentaho Build Process.

We also have CI-dev build server. This is Community-facing build server. This builds only public and open source projects officially called as Community Edition (CE). Runs unit tests, CI creates SNAPSHOT versions. Artifacts created here feed to Pentaho sourceforge page. All the third-party jars are stored in nexus.pentaho.org

- Both DevCI and Release Infrastructure publishes the test result to sonar sonar.pentaho.com, sonar is a unit test server;
- Nexus is a repository which is classified into public (https://nexus.pentaho.org) it is not but an external and Private (https://private.nexus.pentaho.org) is an internal accordingly. This Sonar publishes and resolve the artifacts to both DevCI and Release Infrastructure.

Table below shows what phases are involved in each stage of the build process.

	Local	Wingman	Dev CI	IT/Cerbus	Release/QAT
Unit Tests	✓	✓	✓	✓	✗
Integration Tests	✓	✓	✗	✓	✗
Checkstyle	✓	✓	✗	✓	✗
License header Validation	✓	✓	✗	✗	✗
Test Coverage	✓	✓	✗	✗	✗
Assemblies	✗	✗	✓	✓	✓
Obfuscation	✗	✗	✗	✗	✓
Publish to sonar	✗	✗	✗	✓	✗

Table 6 Build Process.

Pentaho coding stranded

Pentaho adopts coding stranded through the checkstyle, Checkstyle is a development tool to help programmers write Java code that adheres to a coding standard [14]. This allows to check Java code stranded that matches the project. It also finds the problem such as class defined problem, layouts and imports.

Installation Steps for IDE environment:

Step 1. Initially install CheckStyle-IDEA plug-in which is available in repository intelij;

Step 2. Clone https://github.com/pentaho/pentaho-coding-standards repository to concern machine;

Step 3. Check whether the environment directory is set to java 8 JDK (min 1.7 required);

Step 4. Now, configure the CheckStyle plugin with pentaho CheckStyle (File/settings/other settings/checkstyle);

Step 5. Add new CheckStyle link it to the cloned folder (file name pentaho_checkstyle.xml), also have to name property name of your choice and set the property value to "." Now apply changes.

5.3 Sprint work done

This section consists of major work done in most of the sprint, i.e., Maven. We will also see what is Maven, why Maven replaced Ant and a comparison is made on how it is used in Pentaho project and how to build it. The list of tasks that were done during the internship are in Appendix V.

5.3.1 Mavenization

What is Maven?

Maven is a build automation tool used for Java projects. The word Maven means “accumulator of knowledge” in Yiddish. Maven addresses two aspects of building software: first it describes how software builds and second it describes its dependencies [12].

Ant

Ant is an open source Java library provided by “Apache org”. It is a build tool Java applications. Ant provides various built in task such as to build, test, assembly, compile and run an application.

Reasons why Pentaho moved from Ant to Maven

Pentaho java project were ant build, and it took lot of time to compile and build. It also affected the developer’s time while testing the project and creating all targets manually. It was also difficult for new developer to understand the project, because the new developer should do the configuration in build.xml. Later noticing this 21-b product owner tried to build one project and made a comparison with the metrics, there was a big-time variation in generating the artifacts. Later Pentaho has discussion with all developers and decided to move all ant project to maven.

Why to choose Maven over Ant?

In terms of functionality Maven and Ant both are build tools used to ease build process of a project which is provided by Apache org. Table 7 provides the main differences between Ant and Maven.

5.3.2. Maven conversion

There are different ways to configure Maven in Java projects but ideally most things are common, such as Maven use Project Object Model (POM) hierarchy to reduce the duplication that typically exists in Ant projects rather than have each project create an Ant build.xml that takes care of all the details of configuring Sonar properties, such configuration can occur in a Maven parent POM that child projects inherit from [7]. We need to create POM to each module. And for configuration there are several repository managers such as Nexus, Artifactory and Archiva.

Ant	Maven
Ant does not have any formal conventions so we need to provide all in the build.xml file (such as project structure, project goals, target etc.)	Maven has a convention to place source code and compile the code so that we don't need to provide any project structure in the pom.xml (in pom we can configure dependency through dependency management)
Ant is a procedural so we need to provide all information about what and when to do through code also we need to provide order	Maven is simply declarative. we can define everything in the pom.xml file
Since there is not life cycle we need to configure all project goal and target	Maven has its life cycle
Ant is a tool box and it is mainly a build tool	Maven is a framework and it is a project management tool
Ant has to worry transitive dependencies	Maven does not need to worry about transitive dependencies
Ant project differ by build.xml	All maven project has a common structure so that we will get easy understanding
Task is meant to publish the current module descriptor together with its declared publication artifacts to a repository. All the artifacts must have been created *before* calling this task. It does not create the artifacts themselves, but expects to find them at the location indicated by the artifacts pattern [6].	Maven repositories will allow an artifact's to be published alongside the jar of artifact's. Integrated Development Environments (IDE) such as Eclipse allow developers to automatically download the Javadoc and understand how to use the artifact.
Official documentation: http://ant.apache.org	Official documentation: https://maven.apache.org

Table 7 Maven vs Ant.

Below are the steps to migrating from Ant to Maven in terms of Pentaho projects.

Step 1: Identify Maven coordinates of the external dependencies from the nexus repository. We can get it from the existing Ant build also we will have some pile of jars in the lib folder or lib-ivy folder;

Step 2: We need to figure out Maven coordinates for those jars in ant project and we can add it as a dependency in out POM;

Step 3: After identifying these dependencies we need to find a good way where we can share them by setting up a repository manager. Where we have already loaded jars in Pentaho nexus repository (http://nexus.pentaho.org). If not, we have to load it. **Note: Please do not call any external jars using the local path because it will build in local system but that build will break.**

Step 4: We need to create a POM for each module, by actual dependency of each piece of code. For the module, definite structure of Pentaho projects please walk through 5.3.7.

Step 5: For the sample plugin management please refer 5.3.5;

Step 6: After successful completion of all the above steps, we need to work with few files other than POM files. README.md file (update build instruction of the project, test target unit test and integration test, Skiptest if provided), LICENSE.txt (Should include license Apache 2.0, LGPL 2.1, MIT, Mozilla) and .gitignore (where we need to add all files that should be avoided, such as .iml and ide files);

Step 7: Build the project, unzip the project zip folder and compare that it has the same artifact's.

5.3.3 Module structure

i) The primary folder division of a project is done at the root folder by assemblies, api and impl

- assemblies: if the project does any composition of artifacts, then this should be done in projects inside the assembly folder / project;
- api: We should always have an api project separated from the implementation;
- Mavenization process considerations: If there isn't a clear separation of api from implementation, one will not be created during the mavenization process with everything going in the impl folder;
- impl: Where the actual code implementation projects are stored.

ii) We defined the following locations for javascript (code, test, build configuration and test configuration)

- xxx/src/main/javascript
- xxx/src/main/config/javascript/
- xxx/src/test/javascript
- xxx/src/test/config/javascript

iii) I18N should be in:

- xxx/src/main/resources/i18n

iv) Resources

- src/main/resources-filtered
- src/main/resources: These resources are not filtered

v) Obfuscation

- Profile located in pentaho-ee-jar-parent-pom
- Activated by presence of proguard.pro file located at ${basedir}/src/assembly/proguard.pro

vi) OSGI related folders

- src/main/resources/META-INF/js
- src/main/resources/OSGI-INF/blueprint
- assemblies/MyProj/src/main/feature/feature.xml

5.3.4 Dependency Mechanism

Dependency management is a feature in Maven unlike Ant, they are more in concern about using transitive dependency and the property can be activated by <include> or <exclude> function. This would be helpful while handling multi-module projects, something like application that may consist of 100's of module. Dependency management is for centralizing the dependency information as we discussed above like inheriting the parent POM in multi-module projects. The dependency are called using <groupid>, <artifactid>, <version>, <type>, <scope>, <classifier> in the next session.

Transitive Dependencies is a feature that allow user to discover and specify the library and include them automatically.

Example

```
[INFO] +- pentaho-kettle:kettle-engine:jar:8.0-SNAPSHOT:provided
[INFO] |  +- org.pentaho:pdi-engine-api:jar:8.0-SNAPSHOT:provided
[INFO] |  |  \- org.reactivestreams:reactive-streams:jar:1.0.0:provided
[INFO] |  +- pentaho:pentaho-registry:jar:8.0-SNAPSHOT:provided
[INFO] |  +- pentaho:mondrian:jar:3.15-SNAPSHOT:provided
[INFO] |  |  +- xml-apis:xml-apis:jar:2.0.2:provided
[INFO] |  |  +- log4j:log4j:jar:1.2.14:provided
[INFO] |  |  +- commons-math:commons-math:jar:1.1:provided
[INFO] |  |  |  \- commons-discovery:commons-discovery:jar:0.2:provided
```

The above example is pdi-jms-plugin dependency tree, so in this all the dependencies are transitive of kettle-engine dependency, by activating the transitive = true and scope = provided. Let's see the dependency how it is declared in POM of the project. Dependency scope allows user to specify appropriate stage of the build. We will see about the dependency scope in next section. Exclude dependency are the property that can be used to exclude if the downstream dependency is needed.

```
<project>
<parent> </parent>
<properties>
  <kettle.version>8.0-SNAPSHOT</kettle.version>
</properties>
 <dependencyManagement>
  <dependencies>

<dependency>
        <groupId>pentaho-kettle</groupId>
        <artifactId>kettle-engine</artifactId>
        <version>${kettle.version}</version>
        <scope>provided</scope>
</dependency>

  </dependencies>
 </dependencyManagement>
</project>
```

5.3.5 Sample plugin management

```
  <project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                      https://maven.apache.org/xsd/maven-4.0.0.xsd">
  ...
<build>
  ...
  <pluginManagement>
  <plugins>
  <plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-jar-plugin</artifactId>
  <version>2.6</version>
  <executions>
  <execution>
  <id>pre-process-classes</id>
  <phase>compile</phase>
  <goals>
  <goal>jar</goal>
  </goals>
  <configuration>
  <classifier>pre-process</classifier>
  </configuration>
  </execution>
  </executions>
  </plugin>
  </plugins>
  </pluginManagement>
   ...
  </build>
  </project>
```

5.3.6 Building a Pentaho Maven Project

Initial Step: Find the Maven Project by POM in the Project directory.

Pre-requisites for building the project:

- Maven, version 3+;
- Java JDK 1.8;
- This settings.xml in our /.m2 directory, Appendix V.

Building it

Build for nightly/release

All required profiles are activated by the presence of a property named "release".

$ mvn clean install -Drelease

This will build, unit test, and package the whole project (all the sub-modules).

Build for CI/dev

The release builds will compile the source for production (meaning potential obfuscation). To build without that happening, just eliminate the release property.

$ mvn clean install

If the project has the unit test, integration test, the instructions to run the tests are below.

Running the tests

Unit tests

This will run all tests in the project (and sub-modules).

$ mvn test

If the user wants to remote debug a single java unit test (default port is 5005):

$ cd core

$ mvn test -Dtest=<<YourTest>> -Dmaven.surefire.debug

Integration tests

In addition to the unit tests, there are integration tests in the core project.

$ mvn verify -DrunITs

To run a single integration test:

$ mvn verify -DrunITs -Dit.test=<<YourIT>>

To run a single integration test in debug mode (for remote debugging in an IDE) on the default port, 5005

$ mvn verify -DrunITs -Dit.test=<<YourIT>> -Dmaven.failsafe.debug

If the user want to build the project without test

$ mvn clean install -DskipTests can be used - Skip all test

5.3.7 Sample structure of Pentaho Maven Project

Pentaho project has a defined project structure, as shown in Figure 13. This structure is being followed for all Pentaho project so it will be easy to understand for new developer and any Pentaho Developer who work on the project for first time.

```
my-pentaho-project/
├── README.md
├── pom.xml
└── src
    ├── it
    │   ├── java
    │   └── resources
    ├── main
    │   ├── java
    │   │   └── org
    │   │       └── pentaho
    │   │           └── my
    │   │               └── project
    │   │                   └── MyClass.java
    │   └── resources
    └── test
        ├── java
        │   └── org
        │       └── pentaho
        │           └── my
        │               └── project
        │                   └── MyClassTest.java
        └── resources
```

Figure 13 Sample tree structure of a maven project.

5.3.8 Obfuscation for Enterprise Edition Projects

Obfuscation is a process that most of the company adopts to protect intellectual property and to prevent from reverse engineering attacks from hackers. In simple obfuscation is a process of encryption, so Pentaho obfuscates all Enterprise Edition (EE) projects. ProGuard is a code optimizer tool it also reduces the java code size and runs faster. Obfuscation profile is activated by the plugin Obfuscation-project-id.

The file proguard.pro should be located at the right directory *`basedir/src/assembly/……..´*. The dependency is called in the pom.xml as shown.

```
<depemdencyManagment>
 <dependencies>
  <dependency>
     <groupId>pentaho</groupId>
     <artifactId> Obfuscation pro id</artifactId>
     <version>${kettle.version}</version>
     <type>zip</type>
  </dependency>
 </dependencies>
</dependencyManagment>
```

Target to build: *$ mvn clean install –Drelease*

Chapter 6 - Conclusion

From the beginning until the end of Internship It was a great experience in terms of software development, Development of Pentaho software in java with Scrum methodology gave a possibility to learn stranded of coding, now I understood that it is not just important to develop a product and make it work fine it is also important that every other resources (developer) should understand our implementation, algorithm and logics in the program. I have also learned to manage work, task scheduling and interaction with team. I will also say that Pentaho internship was more interesting, every day, because I had an opportunity to know about lot of new bigdata technologies like Spark, MapReduce and Hadoop. I have improved my development skills, Quick learning skills, Time management last but not the least Software Development Process. I have acquired knowledge in build process such as Maven and Ant.

References

[01] Pentaho Licensing Pentaho.iwiki

[02] Schwaber, Ken, and Jeff Sutherland. "The scrum guide." Scrum Alliance (2011)

[03] https://maven.apache.org/what-is-maven.html, accessed on 08-Aug-2017

[04] microstrategy, White paper, Architecture-for-Enterprise-BI, accessed on 08-Aug-2017

[06] http://ant.apache.org/ivy/history/2.2.0/use/publish.html, accessed on 11-Aug-2017

[07] http://www.kyleblaney.com/benefits-of-maven-over-ant, accessed on 15-Aug-2017

[08] https://en.wikibooks.org/wiki/Introduction_to_Software_Engineering/Process/Agile_Model, accessed on 15-Aug-2017

[09] Scrum component diagram by mountaingoatsoftware

[10] http://www.dummies.com/project-management/the-function-of-the-scrum-and-sprint- within-an-agile-project, accessed on 22-Aug-2017

[11] http://wiki.pentaho.com/display/PEOpen/Validating+code+contributions+with+Wingman, accessed on 22-Aug-2017

[12] https://help.pentaho.com/Documentation/7.1/Whats_New, accessed on 22-Aug-2017

[13] Tableau Overview by Vivek Mohan

[14] http://checkstyle.sourceforge.net/, accessed on 28-Aug-2017

[15] iwiki.pentaho, accessed on 20-Sep-2017

[16] https://adtmag.com/articles/2016/10/18/pentaho-platform.aspx, accessed on 08-Aug-2017

[17] https://www.tibco.com/blog/2013/06/28/19781/, accessed on 08-Aug-2017

[18] T. Guarda, M. F. Santos, M. F. Augusto, C. Silva, and F. Pinto, “Process Mining: A framework proposal for Pervasive Business Intelligence,” Inf. Syst. Technol. (CISTI), 2013 8th Iber. Conf., pp. 1 – 4, 2013.

[19] Pentaho Corporation, “Pentaho.” [Online]. Available: http://www.pentaho.com/. [Accessed: 08-Feb-2016]. [9] O. Corporation, “MySQL,” 2016. [Online]. Available: https://www.mysql.com/. [Accessed: 08-Feb-2016]. [10] “MongoDB.” [Online]. Available: https://www.mongodb.org/. [Accessed: 08-Feb-2016]. [11] D. S. Kirschenbaum and G. G. Rosengarten, “Meta-monitoring: Case illustrations of a potential ‘Slump-Buster’ for self-regulatory problems,” J. Clin. Psychol. Med. Settings, vol. 1, no. 3, pp. 245–254, 1994.

[20] A. Marinheiro and J. Bernardino, “Analysis of Open Source Business Intelligence Suites,” Anal. open source Bus. Intell. suites, p. 7, 2013.

[21] M. Golfarelli, “Open Source BI Platforms: a Functional and Architectural Comparison”, DaWaK, pp. 287-297, Springer, 2009.

[22] http://www.jaspersoft.com/, accessed on 20-02-2013

[23] http://www.pentaho.com/, accessed on 20-02-2013

[24] D. J. Power, “A Brief History of Decision Support Systems, version 4.0,” DSSResources.COM. World WideWeb, http://dssresources.com/history/dsshistory.html, version 4.0, March 10, 2007.

[25] PDI training student guide.

Appendices

Appendix I - Pentaho Data Integration – Kettle

This Appendix related to the PDI spoon, PDI is a powerful Extract, transform and load (ETL) solution that uses innovation metadata-driven approach, used to design and build ETL jobs and transformation, which is also results in faster development, low cost maintenance and simple deployment. It has user friendly interface and easy to use in building ETL Jobs and transformation. It is an open source available at *https://github.com/pentaho/pentaho-kettle*.

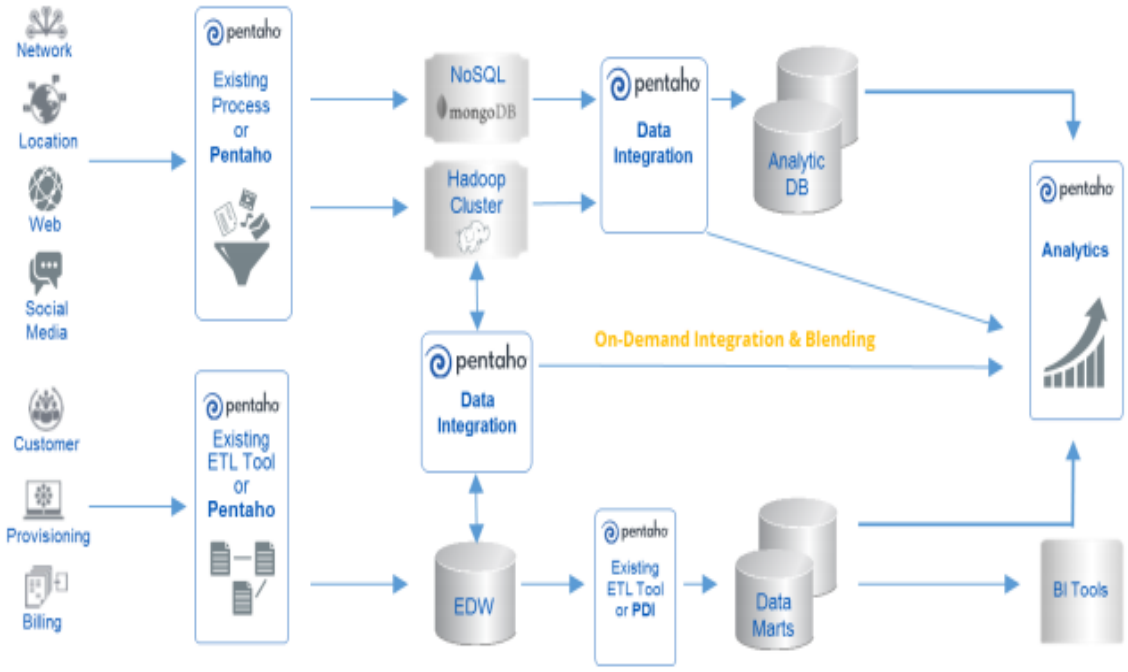

Figure I Components in PDI [25]

PDI consists of several components:

- PDI (aka Kettle) – which consists of these tools. PDI / Kettle is sometimes used to mean Spoon (GUI of PDI / Kettle);
- Spoon – Main GUI which is Graphical job and transformation designer, execute jobs and transformation individually also monitor salve server;
- Carte – it is an HTTP server for remote Execution of jobs and transformation, Cluster with other carte instance to distribute job / Transformation execution;
- Pan – it is a command line execution of Transformations;
- Kitchen – it is a command line execution of jobs;
- Enterprise Edition (EE) Server Data integration - Data Integration is execution of Jobs and Transformations, integrate with existing security e.g. LDAP or Active Directory, Scheduling and Monitoring, Content Management: Managed repository of jobs and transformations.

Diagram below is the core components of PDI

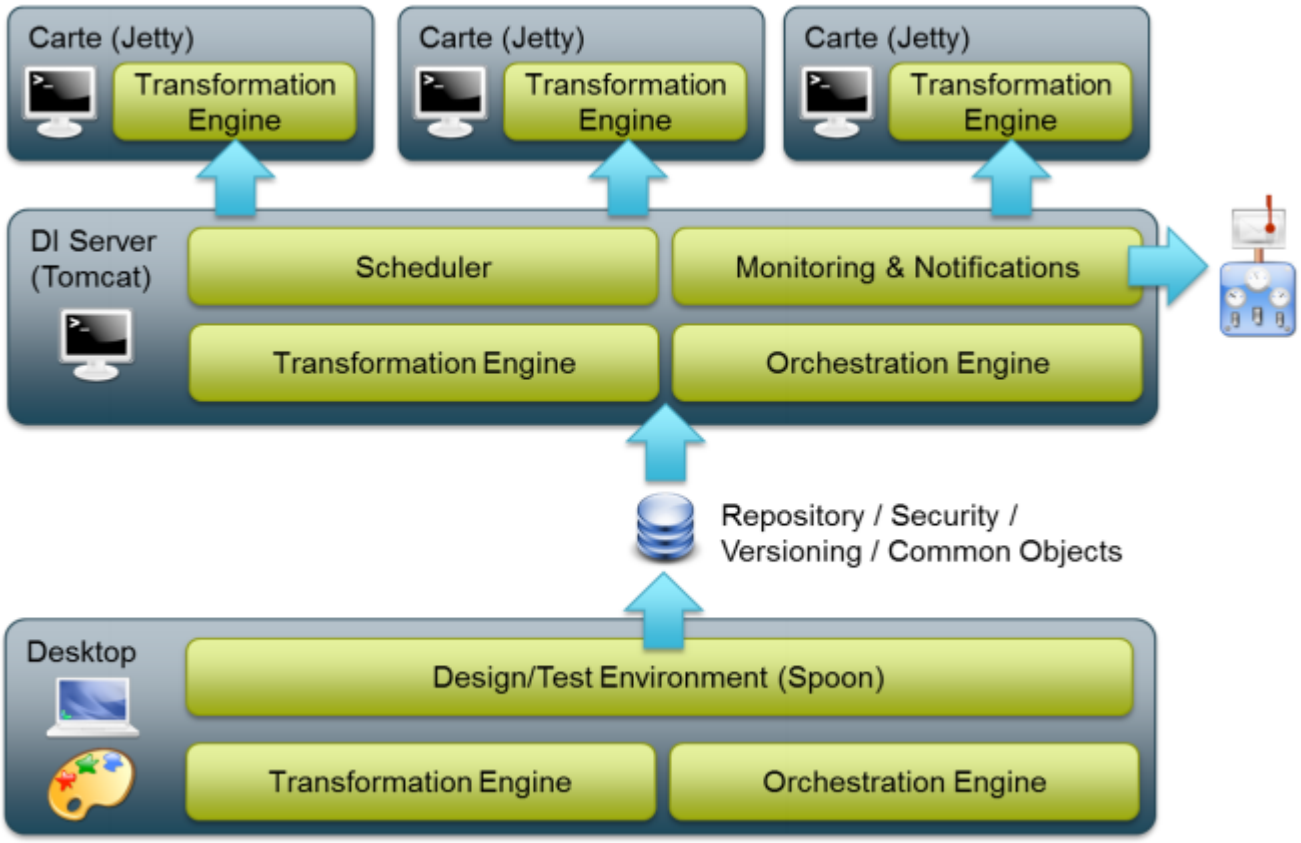

Figure II Core components [25].

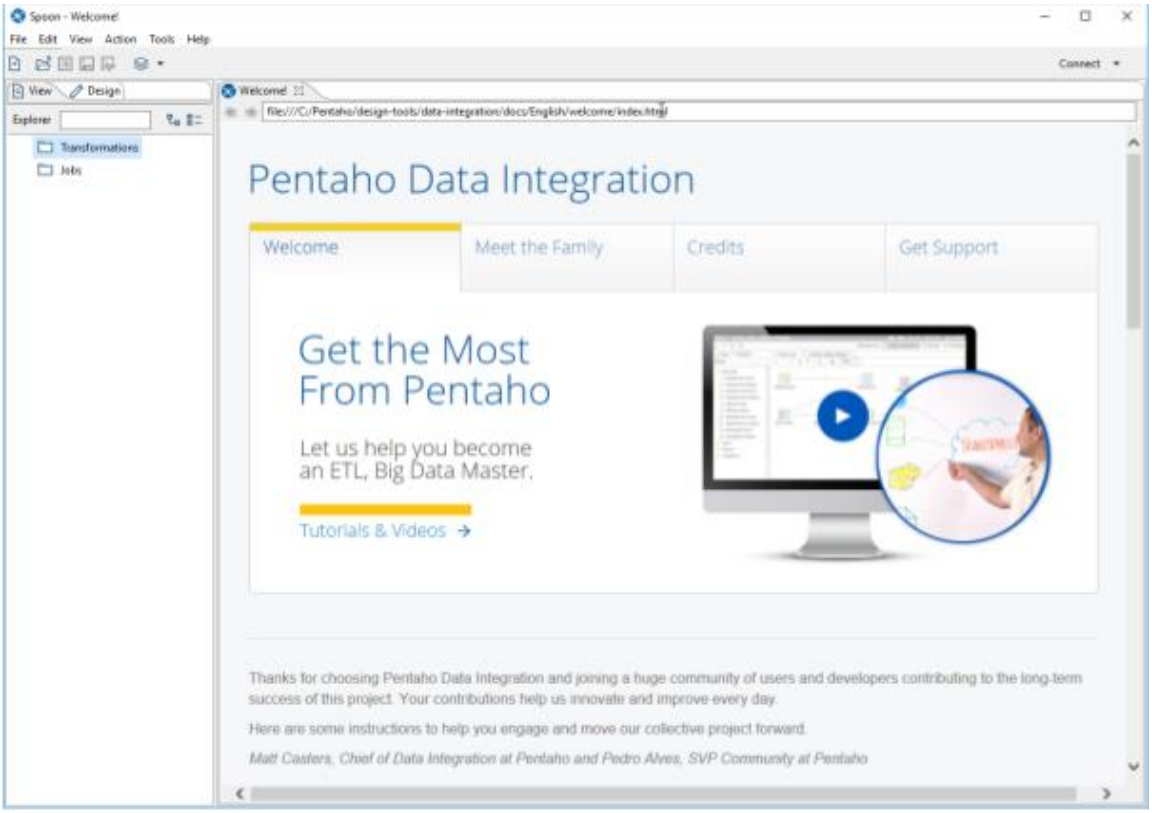

Figure III Spoon Welcome screen.

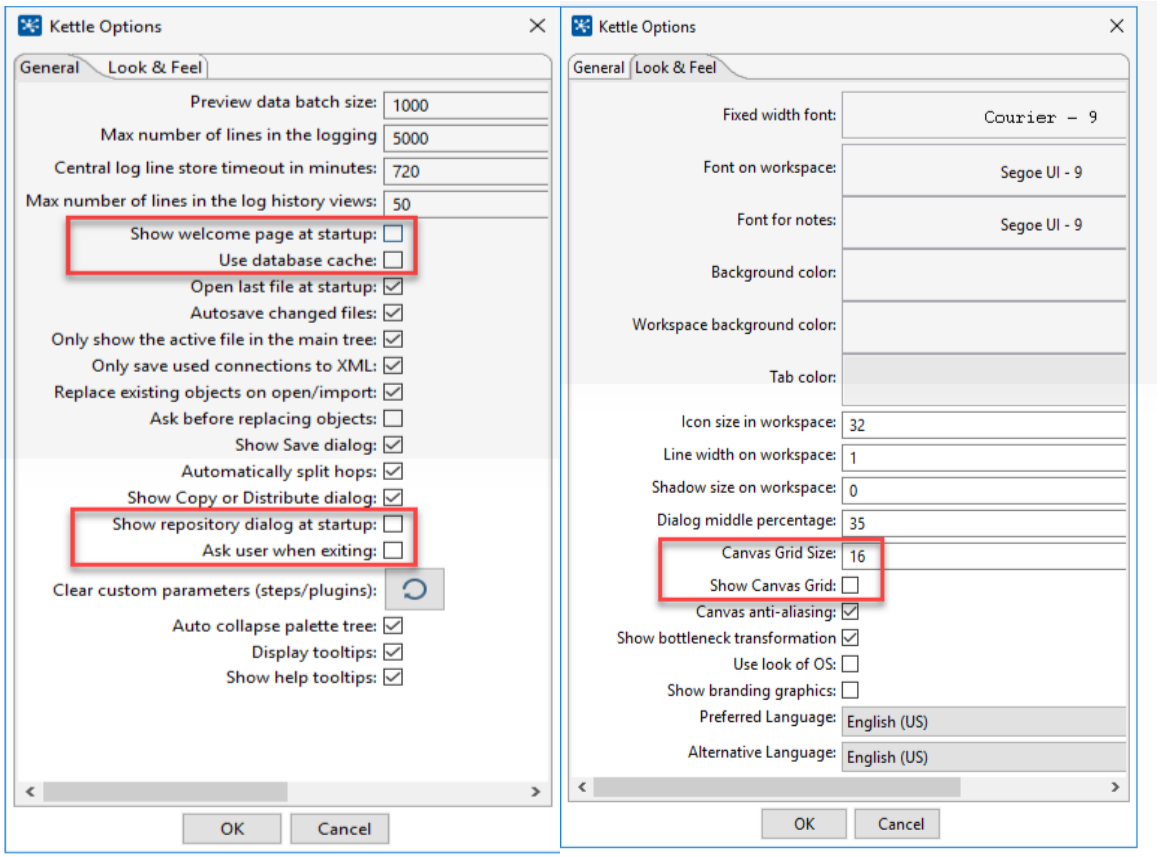

Figure IV Kettle options.

Figure V Pentaho Repository connection.

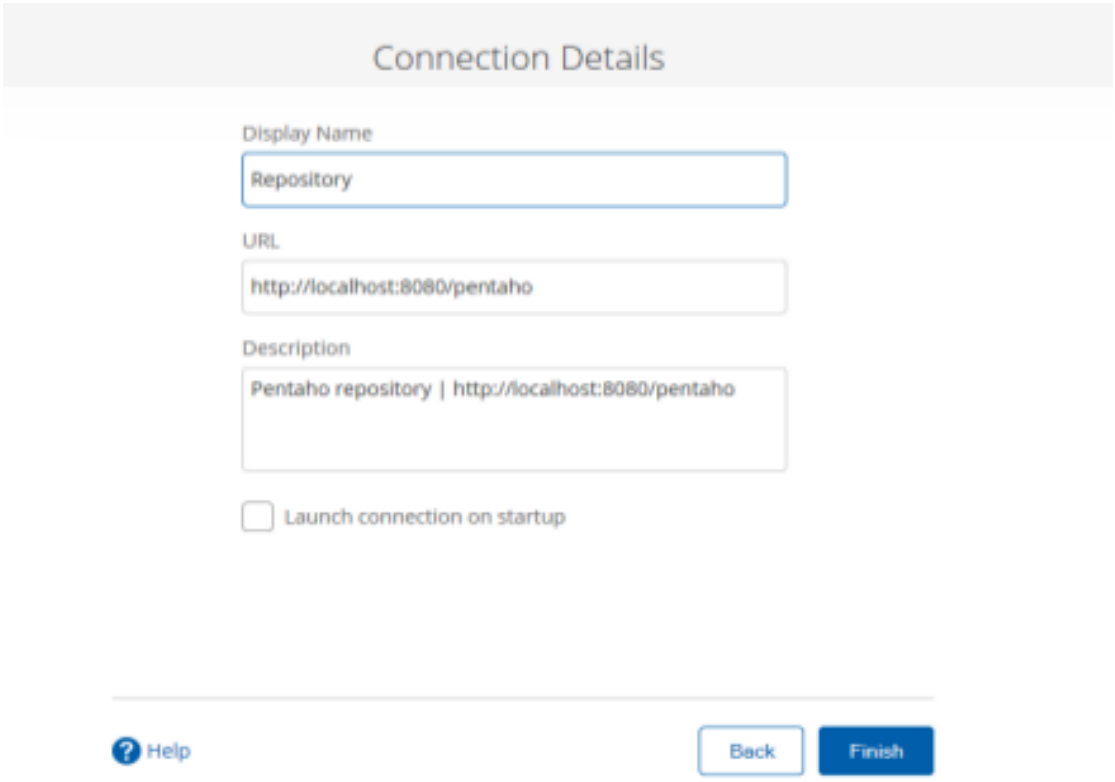

Figure VI Pentaho Repository Connection Screen 2.

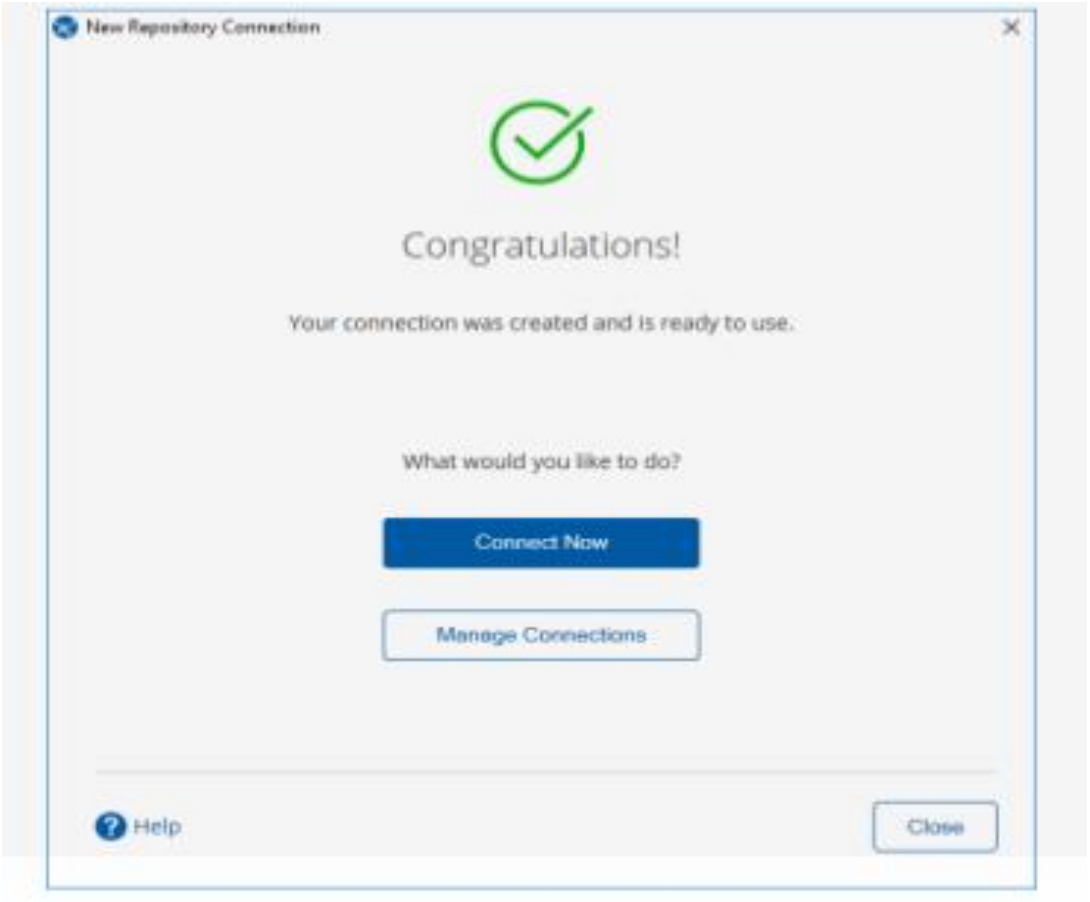

Figure VII Pentaho Repository Connection After connection.

Appendix II – AEL Spark execution

The Appendix is related to the PDI AEL 7.1, the figure below shows step by step execution of Apache Spark.

The main advantage of AEL is that we don't need to worry about the business logic, we can build the transformation in Spoon first, Figure VIII, and then we can choose where and what to execute, Figure IX. We can select Spark engine to execute, Figure X, PDI will send it to the edge node and the driver will then execute. Figure XI successful execution.

AEL spark sample execution screen

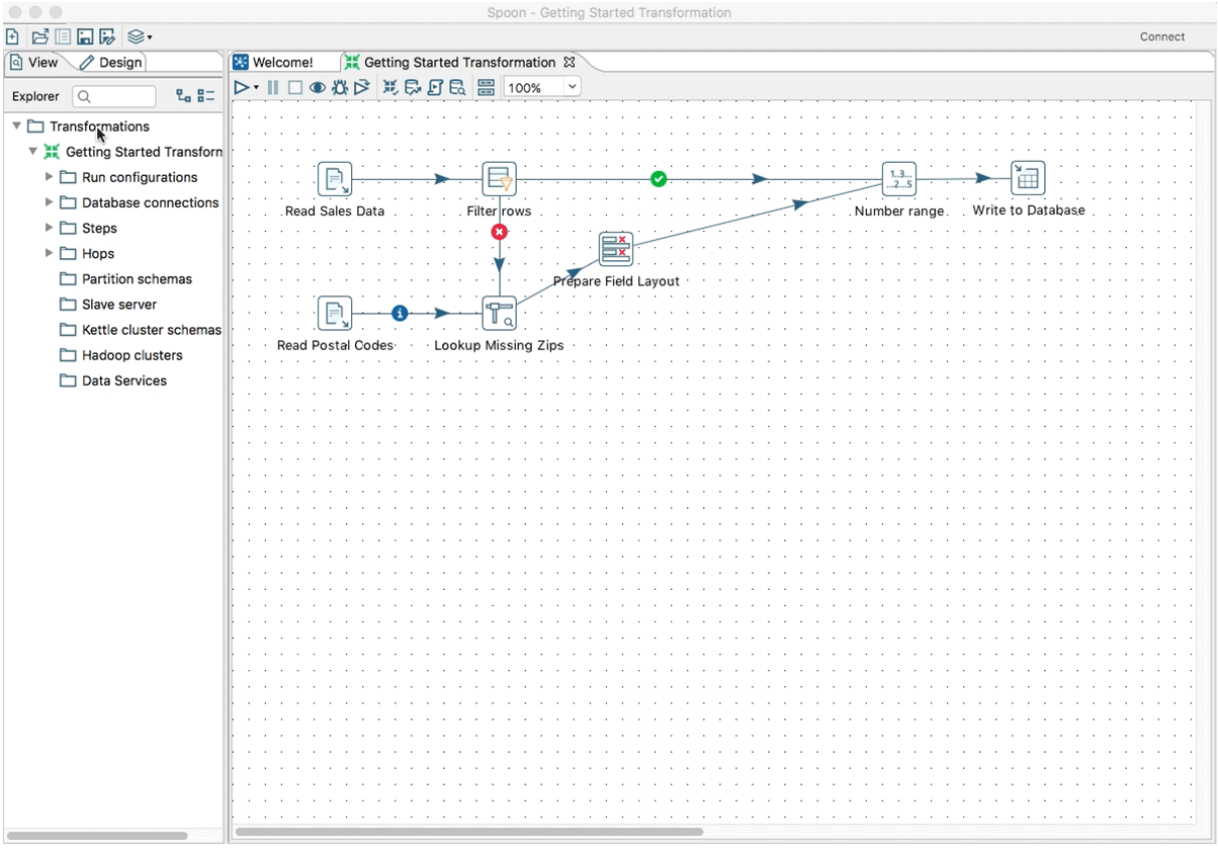

Figure VIII Transformation in spoon Screen 1

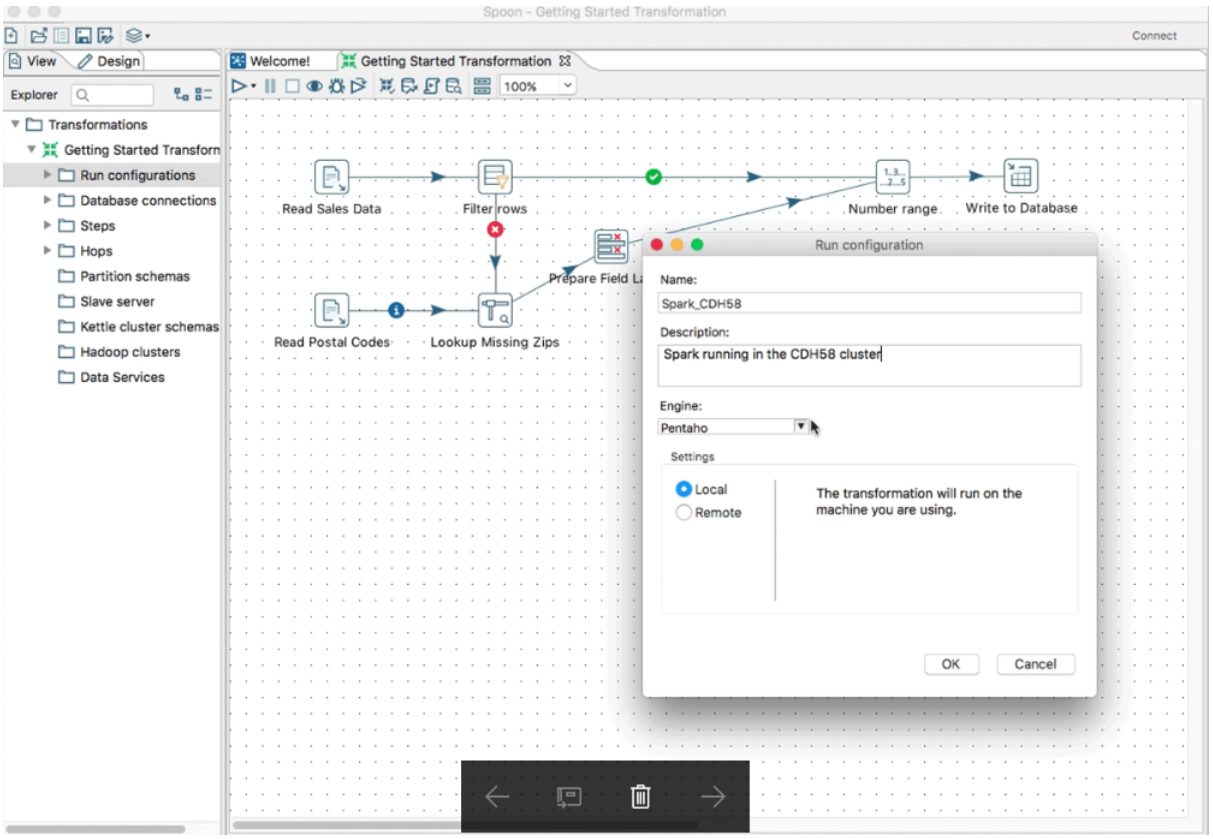

Figure IX Transformation in spoon Screen 2

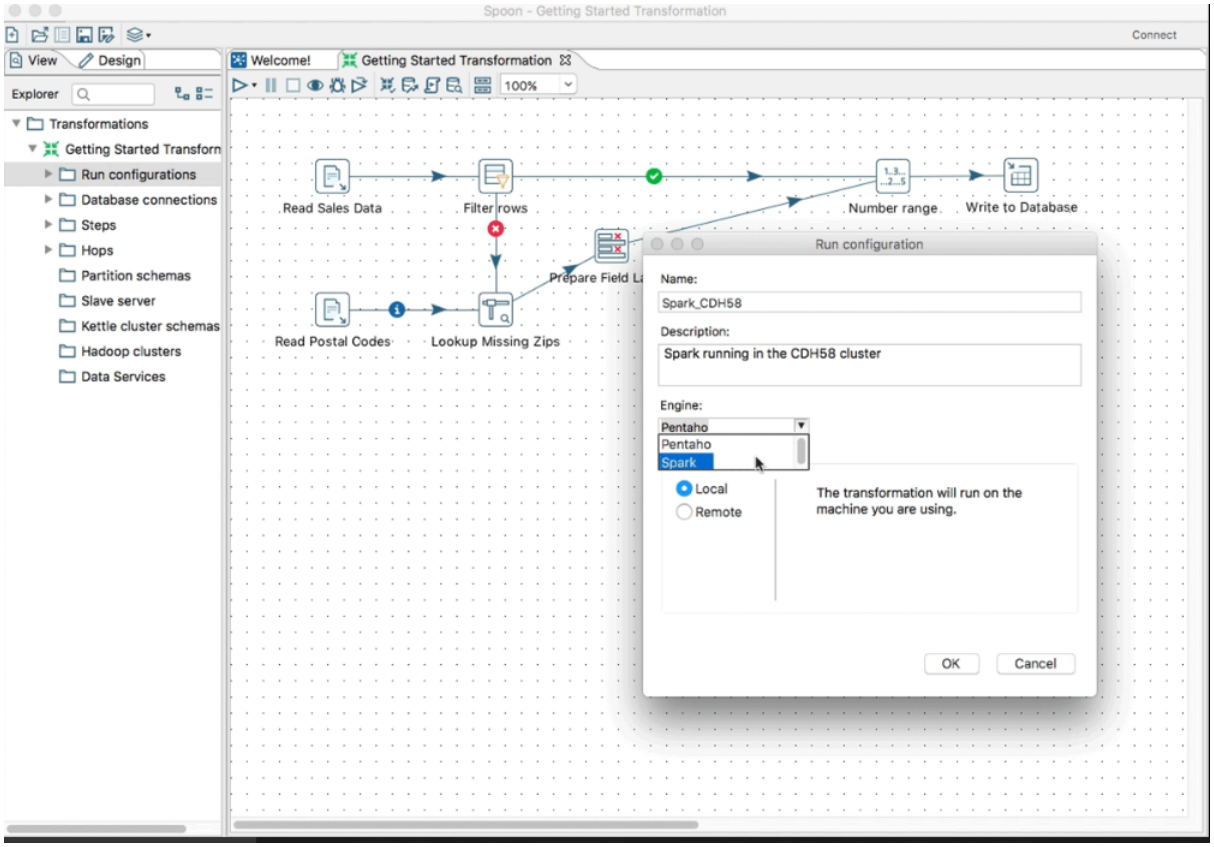

*Figure X Transformation in spoon **Screen 3***

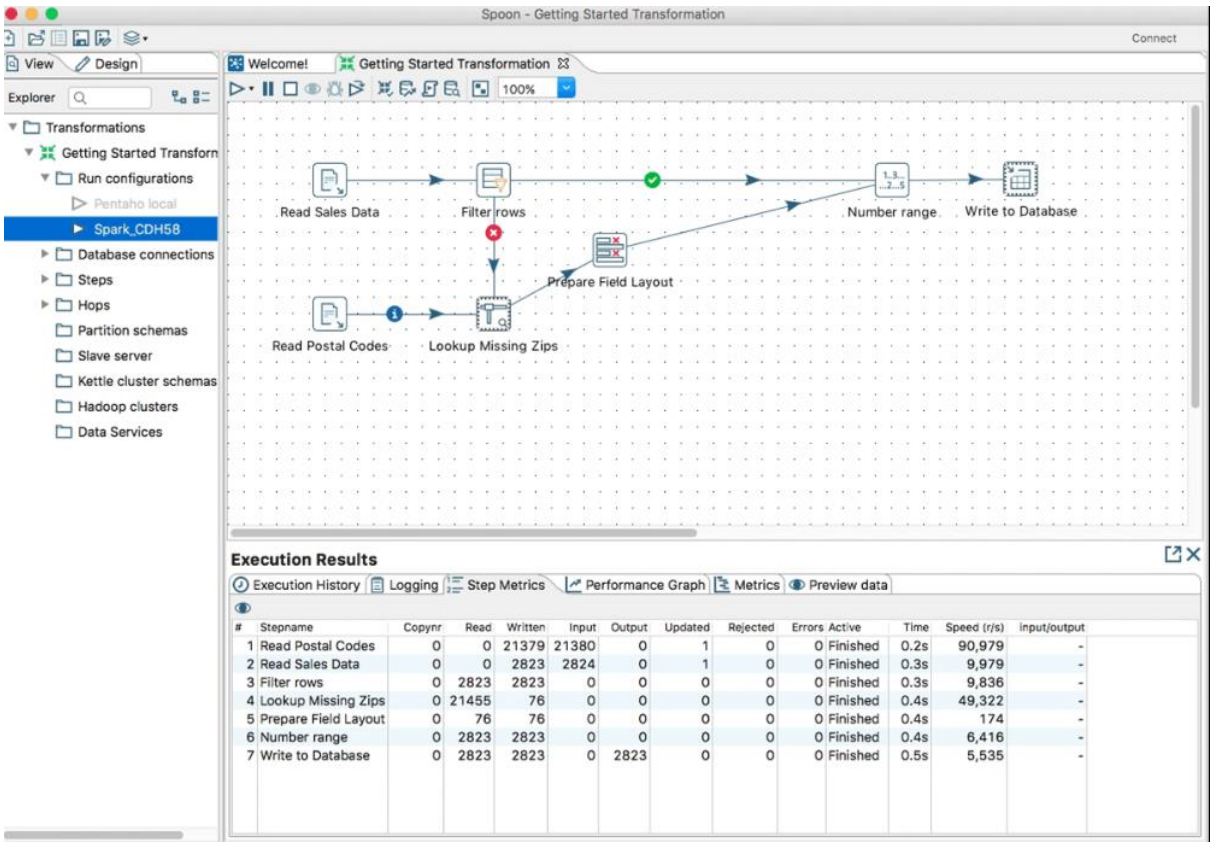

Figure XI Transformation in spoon Screen 4

Appendix III – Sprint tasks

This appendix is the list of all task that are done through the internship. Totally I have participated in fifteen sprint, as listed below.

List of BACKLOGS

Sprint name: **Remember the Fifth**
Sprint started on 2nd feb ends in 16th feb

BACKLOG-12335 Mavenize pentaho-aggdesigner-core
BACKLOG-12725 Mavenize pentaho-aggdesigner-ee
BACKLOG-12918 Mavenize pentaho-aggdesigner-ui
BACKLOG-14008 Mavenize pentaho-vertica-bulkloader
BACKLOG-14394 Modify PDI step archetype to include updated PDI step UX examples
BACKLOG-14414 Pull out the PDI-EE assembly from the platform-ee assembly
BACKLOG-14444 Review shims Mavenization progress
BACKLOG-14445 Review reporting Mavenization progress
BACKLOG-14446 Review common-ui Mavenization progress so far
BACKLOG-14017 Mavenize pentaho-platform-ee
BACKLOG-14268 Fix pentaho-platform CE assemblies to be inline with our maven standards (one assembly per module)

Sprint name: **Revenge of the sixth**
Sprint started on 16th Feb ends on 27th feb

BACKLOG-14122 Test using FindBugs in a Pentaho project
BACKLOG-14393 Update Maven documentation on wiki with recent changes
BACKLOG-14631 Analyze whether or not to add LICENSE.txt to jar artifacts generated by sub-modules which do not explicitly contain one
BACKLOG-14651 Update Mondrian to current Pentaho Maven standards
BACKLOG-14770 Modify PDI job archetype to include updated PDI job UX examples
BACKLOG-14017 Mavenize pentaho-platform-ee
BACKLOG-14268 Fix pentaho-platform CE assemblies to be inline with our maven standards (one assembly per module)
BACKLOG-14769 Review reporting Mavenization progress

Sprint name: **7 of 9**
sprint started on 27th feb end in march 15

BACKLOG-14017 -Mavenize pentaho-platform-ee
BACKLOG-14769 -Review reporting Mavenization progress

BACKLOG-15079 -When opening a report [Public/Steel Wheels/Inventory Report], the report is downloaded instead of being run.
BACKLOG-15104 -PDI About screen showing "Version Unknown"
BACKLOG-13450 -Integration tests fail randomly on common-ui
BACKLOG-14916 -pentaho-analyzer compatibility with 7.1-SNAPSHOT parent pom
BACKLOG-14268 -Fix pentaho-platform CE assemblies to be inline with our maven standards (one assembly per module)
BACKLOG-15131 -Get up to speed on AEL
BACKLOG-15113 -Review AEL Daemon Assembly
BACKLOG-15110 -Investigate the viability of using a Docker Container for local AEL test environment
BACKLOG-15066 -Test using only FindBugs goal in a Pentaho project
BACKLOG-14918 -pentaho-det project compatibility with 7.1-SNAPSHOT parent pom
BACKLOG-14918 -pentaho-det project compatibility with 7.1-SNAPSHOT parent pom
BACKLOG-15111 -Implement IzPack packaging for AEL daemon

Sprint name: **Eight is enough**
sprint starts 16th march ends in 27th march

BACKLOG-15330 - Implement AEL assembly changes
BACKLOG-15354 - Continued updates of AEL Daemon Documentation
BACKLOG-15356 - Continued updates to AEL Docker image project
ENGOPS-2989 - Clean up wrong dependency control properties
BACKLOG-15176 - After switching to maven, reporting project builds complain about test errors
BACKLOG-15329 - Stand up new Wingman node
ENGOPS-2902 - Investigate Sonar report publishing for pentaho-platform-* projects
BACKLOG-14734 - pentaho-i18n-webservice-bundle web resources aren't available

---

Sprint name: **Nein!**
Sprint started on 30th march and ends 12th march

BACKLOG-14057 - Jars checked into source (master)
BACKLOG-15624 - Continued updates of AEL Daemon Documentation
BACKLOG-15625 - Continued updates of AEL Docker image project
BACKLOG-14861 - as a admin, i want to run a script and create a sprak driver to be as light weight as possible
BACKLOG-15663 - Installer should include and deploy PDI AEL daemon assembly archive
BACKLOG-15665 - As an Admin, I would like to use a startup script to invoke and provide parameters to the PDI AEL daemon
BACKLOG-15487 - Research best approach for including LICENSE.txt file in jar artifacts for projects which do not explicitly contain one
BACKLOG-15329 - Standup new wingman node
BACKLOG-15645 - PME - Save domain

BACKLOG-15557 - Update README.md file for all maven projects
BACKLOG-12968 - Review pentaho-kettle/core Maven conversion

Sprint name: **Sprint X**
Sprint started on 13th april and ends on 26th april

BACKLOG-15669 - Update Spark App assembly script with new features
BACKLOG-15665 - As an Admin, I would like to use a startup script to invoke and provide parameters to the PDI AEL daemon
BACKLOG-15329 - Stand up a new Wingman node
BACKLOG-15879 - Use of vunnerable component Apache POI prior to 3.15
BACKLOG-15767 - Update README.md file for pentaho-eula-wrap-config
BACKLOG-15915 - Continued updates of AEL Daemon Documentation
BACKLOG-15916 - Continued updates to AEL Docker Image project
BACKLOG-13026 - Spike investigate potential pentaho obfuscation maven plugin
BACKLOG-15927 - Create Pentaho License plugin
ENGOPS-2518 - Fix/cleanup/improve Nexus Snapshot Plugin code
BACKLOG-13022 - Mavenize pentaho-kettle/plugins/aggregate-rows
BACKLOG-13042 - Mavenize pentaho-kettle/plugins/get-previous-rows
BACKLOG-15893 - Update file "pentaho.jms.cfg"

Sprint name: **11 strange things**
Sprint started on 27/Apr/17 3:25 PM ends on 11/May/17 2:37

BACKLOG-13032 Mavenize pentaho-kettle/plugins/elasticsearch-bulk-insert
BACKLOG-13042 Mavenize pentaho-kettle/plugins/get-previous-row-field
BACKLOG-13069 Mavenize pentaho-kettle/plugins/gp-bulk-loader
BACKLOG-13079 Mavenize pentaho-kettle/plugins/kettle-drools5-plugin
BACKLOG-13089 Mavenize pentaho-kettle/plugins/kettle-dummy-plugin
BACKLOG-13109 * Mavenize pentaho-kettle/plugins/kettle-hl7-plugin
BACKLOG-16168 After installing MQTT Plugin through marketplace and restarting Client, the icon for the step is missing.
BACKLOG-16183 Continued updates of AEL Daemon Documentation
BACKLOG-16184 Continued updates to AEL Docker image project
BACKLOG-16254 * Implement Pentaho obfuscation maven plugin
BACKLOG-16255 * Create parent pom for Pentaho Maven plugins
ENGOPS-3026 SPIKE: Investigate potential Pentaho obfuscation maven plugin

---

Sprint Name: **12 Monkeys**

BACKLOG-12968 Review pentaho-kettle/core Maven conversion
BACKLOG-12977 Review pentaho-kettle/engine Maven conversion

BACKLOG-13004 Review pentaho-kettle/dbdialog Maven conversion
BACKLOG-13143 Mavenize pentaho-kettle/plugins/kettle-s3csvinput-plugin
BACKLOG-13149 Mavenize pentaho-kettle/plugins/kettle-sap-plugin
BACKLOG-13169 Mavenize pentaho-kettle/plugins/kettle-shapefilereader-plugin
BACKLOG-15665 As an Admin, I would like to use a startup script to invoke and provide parameters to the PDI AEL daemon
BACKLOG-15927 Create Pentaho license plugin
BACKLOG-16226 Change the location of the AEL Daemon in the PDI Client Distribution
BACKLOG-16392 AEL - spark app builder doesn't work in windows through Java, requires installer version of pentaho to run
BACKLOG-16400 PRD - Error on start-up: org.pentaho.reporting.designer.core.versionchecker.VersionCheckerUtility
BACKLOG-16409 Continued updates to AEL Docker image project
BACKLOG-16413 AEL - AEL is generally missing from CE pdi-client
BACKLOG-16429 AEL - Daemon startup script issues
BACKLOG-16479 Unit test randomly locked

Sprint name **Baker's dozen**
started on 25/May/17 2:58 PM and ended in 08/Jun/17 2:58 PM

BACKLOG-12293 Create a holistic map of our build processes, tasks, validations
BACKLOG-12968 Review pentaho-kettle/core Maven conversion
BACKLOG-12977 Review pentaho-kettle/engine Maven conversion
BACKLOG-12986 Mavenize pentaho-kettle/assembly
BACKLOG-13004 Review pentaho-kettle/dbdialog Maven conversion
BACKLOG-13013 Review pentaho-kettle/ui-swt Maven conversion
BACKLOG-13059 Mavenize pentaho-kettle/plugins/googleanalytics
BACKLOG-13099 Mavenize pentaho-kettle/plugins/kettle-gpload-plugin
BACKLOG-13115 Mavenize pentaho-kettle/plugins/kettle-json-plugin
BACKLOG-13121 Mavenize pentaho-kettle/plugins/kettle-openerp-plugin
BACKLOG-13281 Mavenize pentaho-kettle/plugins/xml-input-stream
BACKLOG-16752 Update maven-parent-poms documentation

Sprint XIV
Started on 08/jun/17 ended in 22/Jun/17 2:42 PM

BACKLOG-12968 Review pentaho-kettle/core Maven conversion
BACKLOG-12977 Review pentaho-kettle/engine Maven conversion
BACKLOG-12986 Mavenize pentaho-kettle/assembly
BACKLOG-12995 Mavenize pentaho-kettle
BACKLOG-13139 Mavenize pentaho-kettle/plugins/kettle-palo-plugin
BACKLOG-13179 Mavenize pentaho-kettle/plugins/kettle-version-checker
BACKLOG-13189 Mavenize pentaho-kettle/plugins/kettle-xml-plugin

BACKLOG-13199 Mavenize pentaho-kettle/plugins/kettle5-log4j-plugin
BACKLOG-13210 Mavenize pentaho-kettle/plugins/lucid-db-streaming-loader
BACKLOG-13218 Mavenize pentaho-kettle/plugins/ms-access-bulk-loader
BACKLOG-13239 Mavenize pentaho-kettle/plugins/pdi-pur-plugin
BACKLOG-13245 Mavenize pentaho-kettle/plugins/salesforce
BACKLOG-13281 Mavenize pentaho-kettle/plugins/xml-input-stream
BACKLOG-15834 Mavenize pentaho-kettle/plugins/repositories-plugin
BACKLOG-15836 Mavenize pentaho-kettle/plugins/pdi-engine-configuration
BACKLOG-17241 Java unavailable to Wingman docker images
ENGOPS-2932 Put something at least slightly helpful in the CE parent pom project readme

---

Sprint name **Sprint XV**
Started on 22/Jun/17 3:17 PM ended in 06/Jul/17 3:18 PM

BACKLOG-12570 Mavenize jdbc-distribution-utility
BACKLOG-12610 Mavenize pdi-google-docs-plugin
BACKLOG-12618 Mavenize pdi-jms-plugin
BACKLOG-12634 Mavenize pdi-palo-core
BACKLOG-12642 Mavenize pdi-platform-plugin
BACKLOG-12995 Mavenize pentaho-kettle
BACKLOG-15835 Mavenize pentaho-kettle/plugins/meta-inject-plugin
BACKLOG-16218 fix version-merger removing all comments from the pom.xml
BACKLOG-16394 Kettle plugins audit
BACKLOG-17284 Investigate the possibility of no longer generating un-obfuscated EE assemblies
BACKLOG-17285 Investigate the feasibility of creating true snapshot builds
BACKLOG-17292 Review steps in Kettle engine and evaluate difficulty in decoupling
BACKLOG-17327 Update PDI Job Archetype sample dialog to match UX recommendations
BACKLOG-17378 Update licensing and README files for pentaho-kettle and submodules
BACKLOG-17471 Blank Repository Login Screen

Sprint name **Sweet16**
Started on 06/Jul/17 3:18 PM ended in 20/Jul/17 2:34 PM

BACKLOG-12570 Mavenize jdbc-distribution-utility
BACKLOG-12610 Mavenize pdi-google-docs-plugin
BACKLOG-12642 Mavenize pdi-platform-plugin
BACKLOG-16218 fix version-merger removing all comments from the pom.xml
BACKLOG-16256 Investigate long running site phase
BACKLOG-17284 Investigate the possibility of no longer generating un-obfuscated EE assemblies
BACKLOG-17326 Update PDI Step Archetype sample dialog to match UX recommendations
BACKLOG-17327 Update PDI Job Archetype sample dialog to match UX recommendations
BACKLOG-17355 Investigate the interactive reporting project's dependency on dashboards
BACKLOG-17359 Create jobs for building Maven archetypes and plugins

BACKLOG-17424 Wingman building all project instead of only the changed module
BACKLOG-17473 Address any issue which may arise from pentaho-kettle Maven conversion
BACKLOG-17518 Track down all downstream projects with new dependencies introduced by pentaho-kettle Maven conversion

---

Sprint name **17-Almost Legal**
Started on 20/Jul/17 2:53 PM ended in 03/Aug/17 2:38 PM
BACKLOG-12594 Mavenize pdi-ee-plugin
BACKLOG-12610 Mavenize pdi-google-docs-plugin
BACKLOG-12642 Mavenize pdi-platform-plugin
BACKLOG-12660 Mavenize pdi-sap-hana-bulk-loader-plugin
BACKLOG-12669 Mavenize pdi-scheduler-plugin
BACKLOG-12678 Mavenize pdi-teradata-tpt-plugin
BACKLOG-16725 Decouple abort step from Kettle engine
BACKLOG-17022 Multi-module projects can fail compilation due to ordering
BACKLOG-17284 Investigate the possibility of no longer generating un-obfuscated EE assemblies
BACKLOG-17326 Update PDI Step Archetype sample dialog to match UX recommendations
BACKLOG-17327 Update PDI Job Archetype sample dialog to match UX recommendations
BACKLOG-17360 Review job entries in Kettle engine and evaluate difficulty in decoupling
BACKLOG-17424 Wingman building all project instead of only the changed module
BACKLOG-17818 Investigate the best approach to solving the issue with Wingman and multi-module builds
BACKLOG-17829 ClassFormatException occurs during project-info-reports:dependency-convergence execution

---

Sprint name **18- loose lips**
Started on 03/Aug/17 3:25 PM ended in 17/Aug/17 2:38 PM

BACKLOG-12594 Mavenize pdi-ee-plugin
BACKLOG-12669 Mavenize pdi-scheduler-plugin
BACKLOG-12678 Mavenize pdi-teradata-tpt-plugin
BACKLOG-12734 Mavenize pentaho-analysis-ee
BACKLOG-17284 Investigate the possibility of no longer generating un-obfuscated EE assemblies
BACKLOG-17631 Decouple analyticquery step from Kettle engine
BACKLOG-17632 Decouple append step from Kettle engine
BACKLOG-17633 Decouple autodoc step from Kettle engine
BACKLOG-17872 Our default context.xml is using the wrong connection pool factory
BACKLOG-17928 Review and fix mavenized EE projects not obfuscating properly
BACKLOG-18047 change versioning of Mondrian to 8.0

Sprint name **Nifty – 19**
Started on 17/Aug/17 2:58 PM ended in 31/Aug/17 2:34 PM

BACKLOG-12626 Mavenize pdi-operations-mart
BACKLOG-12734 Mavenize pentaho-analysis-ee
BACKLOG-13990 Mavenize pentaho-splunk-plugin
BACKLOG-16230 Remove mongolap and Mondrian 4 dependencies
BACKLOG-16589 Remove Mongolap and Mondrian 4 from release build stack
BACKLOG-17833 Evaluate what is necessary to convert the pdi-ee client project to Maven
BACKLOG-17872 Our default context.xml is using the wrong connection pool factory
BACKLOG-18031 Conduct code reviews for PPP-3487
BACKLOG-18049 Review work done on dependency graph tool so far
BACKLOG-18195 Decouple abort job entry from Kettle engine
BACKLOG-18330 Create xaction component for clearing Mondrian cache and update Ops Mart
BACKLOG-18429 Consolidate decoupled PDI steps into single PDI Core Plugin
BACKLOG-18459 pentaho-server-ee zip does not contain the pentaho-bi-platform-ee obf jar

---

Appendix IV - Executing a PDI transformation

This Appendix show how the execution on PDI loads. When we execute a transformation we typically load the ETL metadata that describes the workload. This is handled by the TransMeta object. Then we execute this workload with a transformation engine handled by the Trans object. This means we first need to load a TransMeta object from XML (with a TransMeta constructor as described below where repository can be null) or from a repository.

```
// Initialize, load settings, plugins, ...
//
KettleEnvironment.init();

...

TransMeta transMeta = new TransMeta("/foo/bar/trans.ktr", repository);
Trans Trans = new Trans(transMeta);

// The following will run the transformation in a separate thread.
//
trans.execute(arguments);

// If you want to wait until the transformation is finished...
//
trans.waitUntilFinished(); //

// If you want to know about the execution result.
//
Result result = trans.getResult();
```

The Result object contains all sorts of interesting results pertaining to the execution, including the number of errors, parsed files and much more as described over here.

Note: Command line arguments are not to be confused with named parameters.

Want to add parameters, variables or arguments to your transformation execution?

```
try {
  for (String key : parameterMap.keySet()) {
    transMeta.setParameterValue(key, parameterMap.get(key));
  }
  for (String key : variableMap.keySet()) {
    transMeta.setVariable(key, variableMap.get(key));
  }
} catch (UnknownParamException e) {
    error(e.getMessage());
}

transMeta.setArguments(arguments);
```

Retrieving data from a step

To retrieve rows of data from any Kettle step copy you can attach a row listener to the step.

```
Trans Trans = new Trans(transMeta);

// prepare the execution of the transformation (instead of simply
execute)
//
trans.prepareExecution(arguments);


// Find a step thread (ready to run but not yet started)
// You can also use method Trans.findBaseStep which gives back a list
of all the step copies
//
StepInterface step = trans.findRunThread("Your Step Name");


// Attach a row listener to a step copy
//
step.addRowListener(new RowAdapter() {

    public void rowReadEvent(RowMetaInterface rowMeta, Object[] row)
throws KettleStepException {
       // Here you get the rows as they are read by the step
    }

    public void rowWrittenEvent(RowMetaInterface rowMeta, Object[]
row) throws KettleStepException {
       // Here you get the rows as they are written by the step
    }
  }
);


// Now start the transformation threads...
//
trans.startThreads();

// If you want to wait until the transformation is finished...
//
trans.waitUntilFinished(); //

// If you want to know about the execution result.
Result result = trans.getResult();
```

Appendix V – Settings.xml

This Appendix contains setting.xml which should be configured in the project/ system to build all Pentaho maven projects.

settings.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
http://maven.apache.org/xsd/settings-1.0.0.xsd">
```

- This is the recommended settings.xml for development of Pentaho projects;
- Off-site community or cloud based builds might want to try running the "central" profile (`-Dcentral`);
- To resolve maven plugin dependencies and transitive's of maven plugins through the maven central repository;
- Instead of the Pentaho repository. This might speed up builds, but is not guaranteed to work for all projects;
- Developer should copy this file to local machine ~/.m2 directory and save it as settings.xml.

This is the mandatory configuration in .m2 on the local machine.

```
<mirrors>
<mirror>
<id>pentaho.resolve.repo</id>
<url>http://nexus.pentaho.org/content/groups/omni</url>
<mirrorOf>!plugins,*</mirrorOf>
</mirror>
</mirrors>

<profiles>
<profile>
<id>pentaho</id>
<activation>
<activeByDefault>true</activeByDefault>
</activation>
<repositories>
<repository>
<id>pentaho.resolve.repo</id>
<name>Pentaho Omni</name>
<url>http://nexus.pentaho.org/content/groups/omni</url>
<releases>
<enabled>true</enabled>
```

```
<updatePolicy>always</updatePolicy>
</releases>
<snapshots>
<enabled>true</enabled>
<updatePolicy>always</updatePolicy>
</snapshots>
</repository>
</repositories>
<pluginRepositories>
<pluginRepository>
<id>plugins</id>
<name>Maven Central</name>
<url>http://nexus.pentaho.org/content/groups/omni</url>
<releases>
<enabled>true</enabled>
<updatePolicy>daily</updatePolicy>
</releases>
<snapshots>
<enabled>true</enabled>
<updatePolicy>always</updatePolicy>
</snapshots>
</pluginRepository>
</pluginRepositories>
</profile>
<profile>
<id>central</id>
<activation>
<activeByDefault>false</activeByDefault>
</activation>
<repositories>
<repository>
<id>pentaho.resolve.repo</id>
<name>Pentaho Omni</name>
<url>http://nexus.pentaho.org/content/groups/omni</url>
<releases>
<enabled>true</enabled>
<updatePolicy>always</updatePolicy>
</releases>
<snapshots>
<enabled>true</enabled>
<updatePolicy>always</updatePolicy>
</snapshots>
</repository>
</repositories>
<pluginRepositories>
<pluginRepository>
<id>plugins</id>
<name>Maven Central</name>
```

```
<url>http://repo1.maven.org/maven2</url>
<releases>
<enabled>true</enabled>
<updatePolicy>daily</updatePolicy>
</releases>
<snapshots>
<enabled>true</enabled>
<updatePolicy>always</updatePolicy>
</snapshots>
</pluginRepository>
</pluginRepositories>
</profile>
</profiles>



<pluginGroups>
<pluginGroup>org.pentaho.maven.plugins</pluginGroup>
</pluginGroups>
</settings>
```

Appendix VI – Matt Advice

This Appendix contain motivation of **Matt Casters** Senior Engineer at Pentaho.

Hi Kettle devs,

As some of you might wonder how Kettle came to be, I wrote this (rather lengthy) historical overview.

As some of you like to point out, in many respects Kettle is not the most beautiful piece of Java code ever written. That is because when I started with Kettle 5 years ago, I had only a little bit of experience with Java 1.1, in writing a Japanese chess database as an aid for my Shogi addiction. (http://http://www.ibridge.be/shogi/)
The ugly piece of code referenced above is the reason why I picked SWT as a GUI framework at a certain time, but that's another story...

In 2001, when the idea for writing my own ETL tool came about, I had been working as a BI consultant for a number of years and found that there was too much messing about in regards to transferring data from one place to another.
As with many software, the driving force behind Kettle simply was: "there has to be a better and cheaper way to do this".
It's one thing to say this. It's a completely different thing to actually write something that is better than inventing ugly data warehouse solutions written in PL/SQL, VB, Shell scripts and what not.
It actually took me 2 years to do a thorough analyses of the problem. Although that might seem a long time, please remember that until last month, I could only work on Kettle during weekends or at night because I had a day time job as a consultant.

In those 2 years, I had written analyses documents and a couple of test-programs in C that used sockets to transfer data. The advantage I saw was that it could work on multiple machines at the same time, etc
The problem with piping was of-course that it was dead slow and that the real problem, extracting data from databases, was not solved at all.

That's why in early 2003 I looked at Java again for the first time in years. A lot of things were moving in the java language, new versions were coming out and more importantly, there were free JDBC drivers to be found for most databases.
Ever so slowly I started working on Value, Row and all the other base classes we have now.
Some time was wasted on trying to write a scripting engine (using my own Byte-code) so that calculations could be done dynamically.

For the archaeologists (or freak show fans) among you: here is the very first archived version of Kettle: http://www.javaforge.com/proj/doc/de...o?doc_id=10680

By mid-2003 building the XML to test quickly became a showstopper. If you have no time to waste, you need to be efficient about it, so I needed a GUI.
SWT was the next big thing and writing Swing, let alone AWT was not my thing. I was using Eclipse to program, and that's all there is to it really.
Although there have been problems with SWT before, it seems to be working fine now for the most part.

The first version of the tool that now is called Spoon, was named "Stir" and at the time and looked like this: http://www.javaforge.com/proj/doc/de...o?doc_id=10682 (It actually looked worse because this is running with the new SWT 3.2 libs)

Stir featured a big X on the graphical view, the log view was not working, neither were most step dialogs, but it might help you understand how the current version came about... or not ;-)
That version can also be found in the archive: http://www.javaforge.com/proj/doc/de...o?doc_id=10683

In the second half of 2003, when the first GUI became available it became less of a burden to write the XML and slowly but surely, things were advancing towards a first version.
Here is a screenshot of 1.0 beta 7 : http://www.javaforge.com/proj/doc/de...o?doc_id=10685 . Even though the number of available steps and supported database has more than doubled since those days, you can see it's starting to look like the version we have now.
The source code of that version is also placed in the archive: http://www.javaforge.com/proj/doc/de...o?doc_id=10686

In 2004 it was working reasonably stable and we I able to deploy it for the first time at a customer. Because of the "real-world" situation, a lot of things needed to be fixed and new features needed to be implemented.
That why in those days, things were advancing a lot faster than the first 3 years. It seemed the code-base grew so fast that several re-factorings and code-cleanings were needed.
Version 2.0. was one of the last "unstructured" versions, to be downloaded here: http://www.javaforge.com/proj/doc/de...o?doc_id=10687

No real java developer would consider NOT using packages in larger projects, and that's exactly what Kettle was slowly but surely becoming.
It was thanks to the Java expertise from companies like ixor (Wim De Clerq especially) that Kettle survived that difficult period.

It was difficult not only because around that period my son Sam was born and little to no time was left for development.
I was also difficult because of the (needless) code complexity. Mostly it was difficult and frustrating because I would be told time and time again by BI colleagues that there were plenty of ETL tools on the market and that it would be completely pointless for me to continue writing one.

Although this has changed "somewhat" since I open sourced Kettle, these kind of remarks are still popping up now and then.
They were (and are) bypassing the simple observation that it should not cost a company thousands of Euros, Dollars or any other currency to do simple things like moving data from one database to another.
When I say cost I'm not only talking about the price of software, but also the time you spend on it. BI consultants like myself (certainly) do no work for free and the longer someone works on a problem the more it costs.

This final observation was key to the success of Kettle because of its simplicity, but it is also a challenge for us as developers as we need to keep looking at the bottom line: an ETL tool helps you do things faster and cheaper.
The end-user requirements are most important and that should remain the case. The Java code was and is nothing more than a way to achieve that goal.

I hope you will find Kettle as fun to work with as I and I hope you found this little stroll into the past interesting, if not you would probably not have read until this line anyway ;-)
Now go back to work! :-) There is plenty of work to do, code to clean and bugs to fix!

All the best,

Matt