INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Reutilização de Processos de Descoberta de Conhecimento

RICARDO FILIPE MARQUES DE SOUSA
Outubro de 2016

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Reutilização de Processos de Descoberta de Conhecimento

Ricardo Filipe Marques Sousa

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Sistemas de Informação e Conhecimento

Orientador: Paulo Oliveira

Coorientador: Paulo Maio

Júri:

Presidente:

[Nome do Presidente, Categoria, Escola]

Vogais:

[Nome do Vogal1, Categoria, Escola]

[Nome do Vogal2, Categoria, Escola] (até 4 vogais)

Porto, outubro de 2016

Resumo

Nesta dissertação é apresentada uma prova de conceito tendo como objetivo automatizar a adaptação de um processo de descoberta de conhecimento previamente definido, a uma nova fonte de dados.

Um processo de descoberta de conhecimento pode usar uma ou mais fonte de dados e, na maioria das vezes necessita, de ajustes quando essas fontes são trocadas por outras pertencentes ao mesmo domínio, uma vez que as fontes de dados anteriores e as novas raramente possuem o mesmo esquema de dados associado. Esses ajustes por vezes fazem com que o analista dispense mais tempo a reformular a configuração do processo implementado anteriormente.

As ferramentas de descoberta de conhecimento existentes permitem que seja executado o processo, mas em nenhuma delas é possível efetuar a troca da fonte de dados e, de uma forma automatizada, criar e ajustar os componentes para que possa ser usado o mesmo processo de descoberta de conhecimento, excetuando os casos em que os esquemas de ambas são rigorosamente iguais. Esta limitação levou à criação de uma solução que permite complementar as ações das ferramentas de descoberta de conhecimento e, desta forma, simplificar a atuação do analista.

A solução desenvolvida foi implementada em C# e permite adaptar o esquema da nova fonte de dados ao processo de descoberta de conhecimento previamente definido, com base na fonte anterior, criando e utilizando os componentes especificados anteriormente. Esta nova solução permite uma maior rapidez no processo de reutilização de um processo de descoberta de conhecimento em novas fontes de dados pertencentes ao mesmo domínio.

Palavras-chave: *Data Mining*, Descoberta de Conhecimento, C#, ClemScript, *IBM SPSS Modeler*, WEKA

Abstract

This thesis presents a proof of concept aiming to automate the adaptation of a process of knowledge discovery previously defined to the new data source.

The Knowledge discovery process can use one or more data sources, and most of time they require adjustments when the data source is exchanged for another one with different schema, but related with the same domain. These adjustments sometimes take a lot of time from analyst due to the need to reshape the process configuration created for the old data source.

The existing tools allow the process to run, but none of them can be used to exchange data source and readjust the process by an automatic method, allowing the creation of new components and redefine the previously implemented components, in order to get the same structure of previous data source. To overcome this limitation, a new solution was developed which not only makes the KDD tools more user friendly but also allows the simplification of the actions needed by analyst.

The developed solution was implemented in C# and allows the new data source to be adjusted to the target schema, based on the previous data source, creating and reusing used previous existing components. This solution allows for greater speed in the adjustment process of the new data source within the same domain.

Keywords: Data Mining, Descoberta de Conhecimento, C#, ClemScript, IBM SPSS Modeler, WEKA

Agradecimentos

Gostaria de agradecer, em primeiro lugar, aos meus orientadores que apoiaram o tema que escolhi para esta dissertação e que sempre me ajudaram quando foi necessário.

Aos professores do mestrado que sempre se disponibilizaram a ajudar, tanto durante as aulas, como durante o desenvolvimento desta dissertação.

Gostaria também de agradecer a toda a estrutura que compõe o Instituto Superior de Engenharia do Porto, que sempre permitiram e forneceram todas as condições necessárias para o desenvolvimento desta dissertação, assim como em especial ao Departamento de Engenharia Informática (DEI).

Um agradecimento muito especial aos meus pais, pela compreensão e pelo apoio prestados durante este período, assim como em relação ao incentivo para que frequentasse o mestrado.

Queria agradecer em especial aos meus colegas de mestrado Bruno da Silva e António Romualdo, pelo companheirismo e pelos trabalhos que realizamos em conjunto.

Por último, e não menos importante, gostava de agradecer aos meus amigos e namorada pela compreensão e pelo apoio prestado.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Códigos

Notação e Glossário

API *Application Programming Interface* – *Interface* de programação de aplicações

CRISP-DM *Cross Industry Standard Process for Data Mining* é uma metodologia criada para o processo de *data mining* que permite interpretar e dividir o mesmo, por forma a torná-lo mais fácil de manter.

Debug Processo de encontrar e eliminar os defeitos que uma aplicação de *software* ou até mesmo o hardware pode conter.

Framework Compreende um conjunto de classes implementadas numa linguagem de programação específica, usadas para auxiliar o desenvolvimento de *software*.

KDD *Knowledge Discovery in Databases.*

Layout É a disposição física dos objetos no espaço.

Plugin Permite adicionar funções a outros programas maiores, provendo alguma funcionalidade especial ou muito específica.

Script Conjunto de instruções para uma determinada ação que o programa ou aplicativo realizará. Um script está sempre associado a uma linguagem, sendo que a sua sintaxe depende dessa linguagem.

SGBD Sistema de gestão de bases de dados

Stream Sequência de objetos que podem ser acedidos por uma determinada ordem

UI *Interface* do utilizador

XML Recomendação da W3C para gerar linguagens de anotação para necessidades especiais.

1 Introdução

Neste capítulo será realizado um enquadramento da importância da reutilização dos processos de descoberta de conhecimento sobre base de dados que se encontram no mesmo domínio de negócio, mas cujo esquema de dados subjacente é diferente. O capítulo inicia com a definição do que são processos de descoberta de conhecimento e a importância dos mesmos.

Segue-se uma caracterização geral do problema, identificando quais os problemas que estão associados quando um analista tenta readaptar um processo de descoberta de conhecimento sobre uma nova fonte de dados, do mesmo domínio, mas com um esquema diferente do esquema original ao qual foi aplicado o processo. A diversidade dos esquemas obriga o analista a despender demasiado tempo no mapeamento e transformação dos dados, sempre que necessita de aplicar novamente o processo de descoberta de conhecimento a um outro conjunto de dados, assim como o tratamento dos dados que cada um contém.

Neste capítulo serão também enunciados os objetivos que se pretendem atingir com o desenvolvimento de uma metodologia que permita dar resposta à necessidade identificada no parágrafo anterior, que foi identificada como sendo comum a todos os analistas de dados.

Por fim, é apresentada a motivação para a realização desta investigação, seguida da forma como se encontra organizado o documento.

1.1 Enquadramento

Cada vez mais as empresas recorrem a analistas de dados para obter o maior conhecimento acerca do seu negócio. Em 1865, Richard Millar Devens deu o primeiro significado ao termo “Business Intelligence”, definindo-o como a maneira como o banqueiro

Sir Henry Furnese compreendeu os problemas políticos, a instabilidade, e o mercado antes dos seus oponentes. Mas, só em 1958, com uma publicação do cientista da computação Hans Peter Luhn, da IBM, é que o BI é reconhecido (Heinze, 2014).

É com recurso a ferramentas de BI que os analistas conseguem extrair o conhecimento presente nos dados. O BI é composto por ferramentas de "online analytical processing" (OLAP) e de descoberta de conhecimento. As ferramentas OLAP estão destinadas a responder a questões colocadas pelos analistas/utilizadores, por sua vez, as de descoberta de conhecimento são ferramentas que permitem extrair conhecimento a partir dos dados, com recurso a um conjunto de técnicas. Para esta dissertação, o foco são estas últimas ferramentas que permitem a aplicação do processo de descoberta de conhecimento ao qual, por vezes, é atribuída a denominação de mineração de dados[1]. Este processo engloba uma seleção prévia de dados, ajustes dos atributos e tipos de dados conforme o/as domínio/necessidades, e a aplicação das técnicas que têm como resultado o conhecimento que possa estar implícito nos dados.

Muitas empresas recorrem a este tipo de serviços para conseguirem extrair conhecimento, por exemplo relativa às vendas ou aos perfis dos seus clientes. Com este conhecimento conseguem criar e promover junto dos clientes os produtos ou serviços que se adequam a cada tipo de cliente, de forma a tornar mais eficiente as promoções e propagandas pretendidas, com base no género de cliente que é habitual adquirir determinada gama de produtos ou serviços, isto é, com base no perfil do cliente.

1.2 Caracterização geral do problema

O termo processo de descoberta de conhecimento[2] (KDD) é proveniente de um workshop de (Fayyad et al. 1996). Esta é uma designação atribuída ao processo de descoberta de conhecimento, que foi definido como:

"O KDD gira em torno da investigação e criação de conhecimento, processos, algoritmos e dos mecanismos para a recuperação de conhecimento potencial em conjuntos de dados" (traduzido de (Norton, 1999)).

[1] Mineração de dados – traduzido do inglês *Data Mining*

[2] Processo de descoberta de conhecimento – traduzido do termo em inglês "Knowledge Discovery in Databases"

O processo de descoberta de conhecimento é constituído por cinco fases: a seleção dos dados, o pré-processamento, a transformação, a mineração dos dados e a avaliação dos resultados. Estas fases encontram-se descritas separadamente na secção "Fases do Processo de Descoberta de Conhecimento em Bases de Dados" que se encontra no segundo capítulo deste documento.

Uma empresa exerce a sua atividade num determinado domínio e recorre a analistas para que estes recolham da fonte de dados conhecimento relevante que possa estar implícito. Mas, se após esta recolha de conhecimento a empresa fornecer novas fontes de dados pertencentes ao mesmo domínio e se quiser que o analista recolha conhecimento presente nestas novas fontes, este terá de implementar novamente ou, pelo menos, readaptar manualmente o processo de descoberta de conhecimento[3] para cada uma, o que fará com que despenda, novamente, tempo a reconfigurar/adaptar o processo implementado anteriormente.

Normalmente, os analistas procuram ajustar/adaptar o pré-processamento e as transformações realizadas à nova fonte de dados que lhes foi fornecida., Esta pode ser proveniente de uma fusão da empresa que os contratou com uma outra empresa do mesmo ramo de negócio e, desta forma, acabando por aplicar transformações adequadas ao domínio em causa, fazendo mapear os atributos do novo esquema a um esquema previamente definido no âmbito do processo de descoberta de conhecimento. É sobre estes ajustes realizados manualmente que se encontra o problema desta dissertação, uma vez que o tempo despendido pelos analistas poderia ser melhor aproveitado.

Segundo alguns autores, "as fases anteriores à mineração de dados consomem a maioria do tempo despendido na análise dos dados" (traduzido de (Perez-Rey, et al., 2006)). Este é um problema comum a todos os analistas, uma vez que procuram adaptar, de forma manual, a nova fonte ao esquema que é criado e esperado na fase de mineração de dados.

1.3 Objetivos deste trabalho

O que se pretende com este trabalho é conseguir disponibilizar uma metodologia de processamento dos dados que poderá vir a servir de suporte ao trabalho desenvolvido pelos

[3] Processo de Descoberta de Conhecimento – do inglês Knowledge Discovery in Databases

analistas aquando da aplicação de um processo de descoberta de conhecimento a uma outra fonte de dados, pertencente ao mesmo domínio.

Deste modo, os objetivos deste trabalho são:

- Desenho de uma nova metodologia para reutilização do processo de descoberta de conhecimento em novas fontes de dados;
- Otimizar o tempo despendido pelos analistas na adaptação de um processo de descoberta de conhecimento;
- Implementação da metodologia numa ferramenta informática;
- Validação da metodologia desenvolvida, mediante a realização de alguns testes/experiências.

Com estes objetivos, é esperado que se consiga colmatar a necessidade apresentada e simplificar o processo, tanto na seleção como na transformação dos dados.

1.4 Abordagem preconizada

Sendo um problema relacionado com descoberta de conhecimento e o modo como relacionar fontes de dados do mesmo domínio, mas com os dados segundo esquemas diferentes, houve a necessidade de identificar um ponto no processo de descoberta de conhecimento que permitisse obter um esquema de dados que seria o ponto de ligação entre a fase de transformação e a de mineração de dados. Isto é, identificar no processo de descoberta de conhecimento onde seria possível definir uma metodologia para simplificar as ações que geralmente são tomadas manualmente pelo analista, partindo de um novo esquema e gerando o esquema esperado pré-mineração de dados. Esse esquema será gerado a partir da fonte de dados inicial e permitirá simplificar o mapeamento da segunda fonte, para que seja possível efetuar todas as operações necessárias, de modo a obter o mesmo esquema pré-mineração.

Desta forma, a abordagem centra-se no mapeamento entre a nova fonte e o esquema de dados esperado, assim como entre as fontes de dados, permitindo a identificação dos atributos em falta e das possíveis transformações necessárias.

1.5 Resultados Obtidos

Perante o problema da reutilização do processo de descoberta de conhecimento, a solução desenhada consiste numa metodologia que se baseia nas ações que o analista toma e que se ajusta às ferramentas de descoberta de conhecimento existentes, sendo que estas são apresentadas na Secção 2.5 - "Ferramentas de Descoberta de Conhecimento". Essas ferramentas disponibilizam componentes que permitem ao utilizador manipular os dados e os atributos que considerar relevantes para o processo de descoberta de conhecimento.

A abordagem adotada permite, de uma forma automatizada, gerar os componentes específicos de cada ferramenta e que se ajustam a cada uma das situações, para que a nova fonte de dados seja transformada e corresponda ao esquema pré-mineração que é esperado. A gestão desses mesmos componentes é efetuada com base nos mapeamentos entre as fontes e o esquema de dados pré-mineração, realizando assim as operações das fases de pré-processamento e transformação dos dados consideradas relevantes e necessárias.

1.6 Estrutura do documento

Este documento contém a seguinte estrutura: Introdução, Contexto e Estado da Arte, Solução proposta, Testes, Experiências e Avaliação da Solução, e Conclusão.

A Introdução é um capítulo destinado a apresentar muito sucintamente o objetivo desta dissertação e dar um enquadramento do tema da mesma.

O capítulo Contexto e Estado da Arte tem como objetivo apresentar o processo de descoberta de conhecimento por forma a percebê-lo melhor e a identificar os pontos de melhoria que este poderá sofrer, assim como as ferramentas que permitem a implementação deste processo.

O capítulo seguinte terá por base a possível solução selecionada e descreverá o todo trabalho realizado na sua implementação, relatando deste modo as opções tomadas e limitações encontradas e como foram ultrapassadas.

No capítulo Testes são apresentados os casos de teste funcionais e os testes unitários aos desenvolvimentos, por forma a validar a implementação efetuada, referindo a cobertura dos mesmos.

Por sua vez, no capítulo Experimentação e Avaliação da Solução são apresentadas as experiencias efetuadas para que estas evidenciem os possíveis ganhos com a solução a ser construída.

Por último, a Conclusão que tem como objetivo apresentar quais foram as aprendizagens com a realização desta dissertação, a evidência dos objetivos alcançados, as limitações da solução e o trabalho futuro a ser realizado.

2 Contexto e Estado da Arte

A descoberta de conhecimento, como já foi referido anteriormente, é um processo executado sobre um ou mais esquemas de dados, permitindo extrair padrões presentes nos mesmos, classificando assim determinados tipos de ações ou atividades que se foram registando ao longo do período em análise.

Nos dias de hoje, o processo de descoberta de conhecimento ainda exige que o analista defina todas as transformações necessárias nos dados, para que consiga extrair o máximo de conhecimento possível, tendo que recomeçar todo este processo sempre que o pretender aplicar a uma outra fonte de dados, embora do mesmo domínio, por possuir um esquema diferente. O recomeçar significa obrigar a que este despenda tempo no mapeamento dos novos atributos e a aplicar os ajustes que considerar necessários ao processo anteriormente definido. A realização de operações de mineração sobre mais do que uma fonte acarreta problemas (Zhang, et al., 2003). Segundo este autor, a realização destas operações sobre múltiplas fontes de dados do mesmo domínio deverá ser feita a nível local (Zhang, et al., 2003). Isto é, deverão ser criados processos de descoberta de conhecimento isoladamente para cada fonte e não serem feitas análises conjuntas à globalidade dos dados.

Neste capítulo é feita uma descrição acerca do estado da arte em relação à descoberta de conhecimento, descrevendo o foco principal da dissertação, como se enquadra no contexto da descoberta de conhecimento, qual será a metodologia a seguir para atingir os objetivos previamente estabelecidos e, por fim, quais as tecnologias que irão ser utilizadas para atingir os mesmos.

2.1 Processo de descoberta de conhecimento

O processo de descoberta de conhecimento em bases de dados engloba diversas áreas científicas, incluindo estatística, ciência da computação, e negócio, bem como um conjunto de metodologias. Metodologias essas que advêm da aprendizagem máquina, reconhecimento de padrões, inteligência artificial, aquisição de conhecimento para sistemas inteligentes, entre outras.

2.1.1 Definição de Descoberta de Conhecimento

A descoberta do conhecimento é definida como uma área interdisciplinar que incide sobre metodologias de extração de conhecimento útil a partir dos dados. Facultado um conjunto de factos (F), uma linguagem (L), e algumas medidas de certeza (C), permitem definir que um padrão é uma frase (S) na L que descreve relações entre um subconjunto Fs de F. Desta forma, conhecimento é um padrão que é interessante e certo o suficiente conforme os critérios do utilizador.

Muitos padrões são extraídos das bases de dados, mas apenas alguns desses é que são considerados interessantes, e úteis, sendo assim considerados conhecimento. “Um conhecimento é útil quando ajuda a alcançar um objetivo do sistema ou do utilizador (Frawley, et al., 1991)”.

A saída de um programa que monitoriza o conjunto de factos numa base de dados e produz padrões neste sentido é descoberta de conhecimento. É com base nesses factos presentes numa base de dados que é possível extrair o conhecimento que, por vezes, se encontra implícito, ou que permite traçar perfis/padrões nos dados.

2.1.1.1 Padrões e Linguagens

Quando é referido padrão ou linguagem, deverá ser entendido como sendo algo que é expresso numa linguagem de alto-nível, isto é, que é percetível pelo humano, como por exemplo:

```
Se Idade < 25 e Instrutor de condução = Não
        Então Culpado em caso de acidente = Sim

Com probabilidade entre 0.2 e 0.3
```

Código 1 - Linguagem alto nível

Neste caso, o padrão encontrado está descrito numa linguagem de alto-nível, percetível pelo ser humano. Estes padrões podem ser usados pelas pessoas ou como parâmetros de entrada para outros sistemas, tais como: sistemas de apoio à decisão (Frawley, et al., 1991).

2.1.1.2 Certeza

Representar e transmitir o grau de certeza é essencial para determinar a quantidade de confiança que o sistema ou o utilizador deverá depositar sobre uma determinada descoberta. Mas a certeza envolve muitos fatores, incluindo: a integridade dos dados; o tamanho da amostra para extração do conhecimento; e o grau de apoio a partir do conhecimento de domínio disponível. O utilizador é, por norma, quem estipula qual o grau de certeza mínimo que um determinado padrão, identificado pelo algoritmo, tem de ter para que esse não seja descartado (Frawley, et al., 1991).

2.1.1.3 Interesse

Os padrões também têm de ter associado um nível de interesse considerado razoável para que sejam considerados conhecimento. Estes são interessantes quando são novos e úteis. Importa referir que são novos dependendo da estrutura assumida de referência, que pode ser no âmbito do sistema ou do utilizador. Já a utilidade de um padrão está relacionado com o facto de este poder ajudar a alcançar um objetivo do sistema ou do utilizador.

Nem todos os padrões presentes num esquema de dados são considerados interessantes, mesmo sendo considerados novos e úteis, já que, estes são triviais para cálculos, e para serem não triviais, um sistema deve fazer mais do que cálculos estatísticos com eles.

Um sistema de descoberta deve ser capaz de decidir que cálculos realizar e se os resultados são interessantes o suficiente para constituir conhecimento no contexto atual. Outro ponto de vista desta noção de não triviais é o de que um sistema de descoberta deve

processar com algum grau de autonomia os dados e a avaliação dos mesmos (Frawley, et al., 1991). b

2.1.1.4 Eficiência

A eficiência de um processo de descoberta de conhecimento está na eficiência do algoritmo escolhido, sendo que este algoritmo só é considerado eficiente se o tempo de execução e espaço utilizados seguirem uma função polinomial de baixo grau do comprimento de entrada. Estudos anteriores na teoria de aprendizagem computacional (Valtant, 1984; Haussler, 1988) mostraram que não é possível aprender eficientemente um conceito *Boolean* arbitrário, e isto é considerado de complexidade NP-difícil[4]. Contudo estes resultados estão normalmente relacionados com o pior caso de desempenho de algoritmos e não é impeditivo de se encontrar conceitos complexos mais rapidamente.

Existe a possibilidade de abandonar a procura de um algoritmo que aprenda um conceito desejado com alguma garantia e, ao invés disso, aceitar algoritmos de heurísticas e de aproximação (Frawley, et al., 1991).

2.1.2 Fases do Processo de Descoberta de Conhecimento em Bases de Dados

O processo de descoberta de conhecimento é composto por cinco fases: seleção dos dados, processamento, transformação, mineração de dados e a interpretação/avaliação dos resultados (Fayyad, et al., 1996).

Este processo é um processo iterativo, ou seja, é um processo que, quando se encontra na última fase, pode voltar a uma das fases anteriores, usando já o novo conhecimento adquirido, caso assim o seja indicado. Na Figura 1 estão ilustradas as fases do processo de descoberta de conhecimento e como estas se interligam.

[4] NP-difícil – em teoria da complexidade computacional, é uma classe de problemas que podem ser: problemas de decisão, problemas de pesquisa ou problemas de otimização.

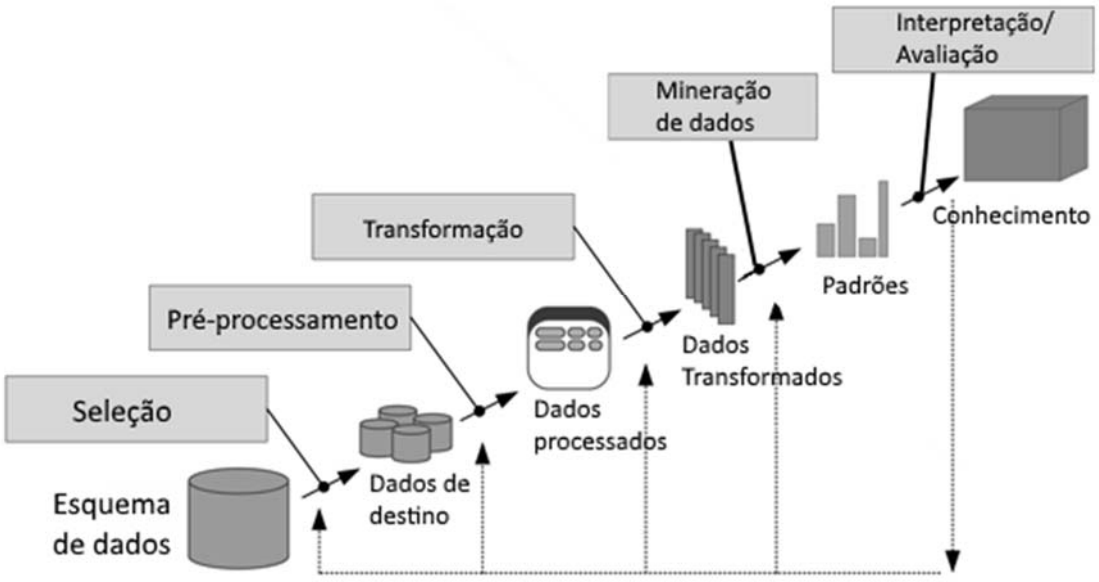

Figura 1 – Fases do processo de descoberta de conhecimento

2.1.2.1 Seleção

A fase da seleção, tal como referido anteriormente, é a primeira do processo de descoberta de conhecimento. Esta consiste em recolher/selecionar os dados significantes para a descoberta de conhecimento, portanto, trata-se de selecionar uma amostra dos dados sobre a qual serão aplicadas transformações e algoritmos de mineração (Fayyad, et al., 1996).

"As decisões primeiro devem ser tomadas em relação à natureza do problema em causa, por forma a avaliar a sua adequabilidade ao processo de descoberta de conhecimento em base de dados." (traduzido de (Fayyad, et al., 1996)). Nem todos os problemas se adequam a 100% ao KDD, existem os que se adequam mais ou menos, e os que não fazem sentido em KDD, isto é, por vezes existem problemas em que o objetivo não é facilmente atingível, já que poderá envolver regras de negócio e que estão muito dependentes de cada situação, apenas é possível extrair uma possível solução (Lodder, et al., 2006).

Assim sendo, esta fase consiste em criar uma amostragem dos dados existentes, selecionando apenas parte dos dados da fonte (Marsh, 2014).

2.1.2.2 Pré-processamento

O pré-processamento é a fase de limpeza e preparação dos dados. É nesta fase que a fiabilidade destes é aumentada, aplicando-lhes alguns filtros por forma a determinar os valores em falta, ou remover valores extremos/desviantes (valores que se distanciam dos

valores considerados “aceitáveis”, ou que não se enquadram num determinado intervalo de valores).

Para que esta fase seja aplicada pode ser necessário recorrer a algoritmos de mineração de dados por forma a prever quais serão os valores em falta e ter esses campos como objetivo de um algoritmo supervisionado de mineração de dados. A previsão dos valores em falta é feita com base no modelo desenvolvido para esse atributo (Marsh, 2014).

Esta é umas das fases em que o utilizador é considerado uma peça fundamental, visto que este poderá influenciar nos algoritmos ou nas tarefas a serem escolhidas.

2.1.2.3 Transformação de dados

Esta é a fase em que antecede a fase da mineração de dados. Nesta fase são gerados dados uniformizados, que seguem uma determinada nomenclatura ou terminologia. Esses dados podem melhorar o conhecimento que se pode vir a obter posteriormente na última fase do KDD, como descrito a seguir.

Esta é uma fase crucial para o sucesso do processo de descoberta de conhecimento, e é normalmente configurada para um projeto muito específico (Marsh, 2014).

2.1.2.1 Mineração de dados

A fase de mineração de dados é apenas uma das fases do processo de descoberta de conhecimento, apesar de por vezes, ser atribuído este nome ao processo. No entanto, isso acontece porque esta é a fase mais importante deste processo, como tal, é usado como sinónimo.

A fase da mineração de dados é a fase em que é necessário definir qual o tipo de algoritmo de mineração a ser usado, isto é, se irão ser adotados algoritmos de classificação, regressão ou de agrupamento[5], entre outros, uma vez que está dependente dos objetivos do processo de descoberta de conhecimento e do tipo de conhecimento que se pretende obter.

Por vezes, existe a necessidade de executar uma tarefa de mineração por forma a obter uma previsão para determinado atributo conforme determinados critérios, ou então, também surge a necessidade de obter uma descrição dos mesmos. Ou seja, por exemplo, perceber o porquê de determinados clientes optarem por determinados produtos ao invés de outros. Sendo que a previsão está mais associada a algoritmos supervisionados, enquanto os

[5] Agrupamento – traduzido do termo em inglês Clustering

de descrição incluem algoritmos não supervisionados e visualização de determinados aspetos de mineração de dados.

A maioria das técnicas de mineração de dados são baseadas em aprendizagem indutiva, onde um modelo é construído implicitamente pela generalização a partir de uns exemplos de treino, sendo que, uma abordagem indutiva permite que o modelo treinado possa ser usado numa abordagem futura.

Posteriormente é aplicado o algoritmo e é executado o número de vezes necessários até que o resultado esperado seja obtido, ajustando os parâmetros de controlo do mesmo (Marsh, 2014).

A mineração de dados inclui a pesquisa de padrões de interesse num conjunto de representações através de regras ou árvores de classificação, regressão, modelação sequencial, dependência, e análises de linha[6] (Fayyad, et al., 1996).

2.1.2.2 Interpretação/Avaliação

Finalmente, depois de serem aplicados os padrões extraídos através do algoritmo escolhido, estes são analisados e interpretados, tendo em conta os objetivos que se pretendia alcançar. É nesta fase que o conhecimento extraído é documentado e arquivado por forma a ser usado numa posterior utilização (Marsh, 2014).

É também nesta fase que são interpretados os resultados obtidos, e em que o processo poderá retroceder alguns passos atrás até serem obtidos os padrões, e estes serem marcados como redundantes ou irrelevantes quando não satisfaçam os critérios do utilizador (Fayyad, et al., 1996).

Esta é outra das fases em que o utilizador, agente especializado, tem o poder de decidir se aceita os conjuntos de dados e padrões extraídos e, desta forma, poderá alterar os parâmetros dos algoritmos utilizados provocando assim uma nova iteração.

[6] Análises de linha – traduzido do termo inglês line analysis

2.1.3 *Framework* para a Descoberta de Conhecimento

Embora os sistemas de descoberta possam variar consideravelmente na forma como são concebidos, na Figura 2, estão ilustrados os elementos base para a constituição de um protótipo de um sistema de descoberta de conhecimento.

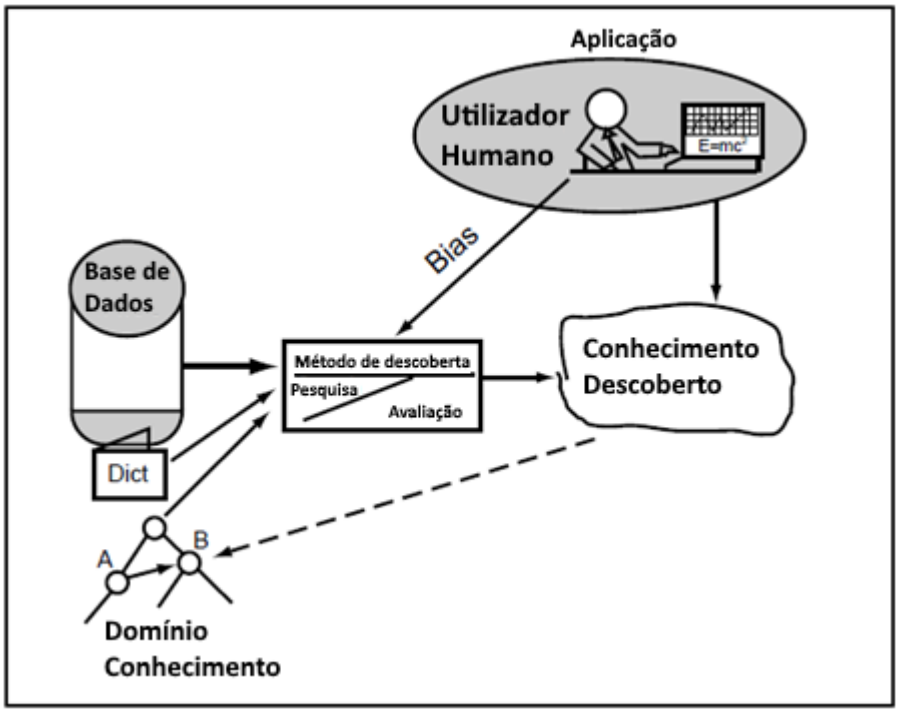

Figura 2 – Framework para descoberta de conhecimento em base de dados (Frawley, et al., 1991)

Como se encontra ilustrado na Figura 2, existem componentes necessários na constituição de um sistema de descoberta de conhecimento, sendo eles: o(s) algoritmo(s) de descoberta, os dados ainda não manipulados e o conhecimento que se consegue extrair desses dados.

No núcleo de um sistema de descoberta de conhecimento está o método de descoberta, que processa e avalia os padrões que encontra tornando-os assim em conhecimento. Tendo como informação inicial: os dados provenientes de fontes de dados, o dicionário de dados, o conhecimento adicional relativo ao domínio em questão e as configurações do utilizador. Por sua vez os resultados podem ser usados pelo utilizador, ou serem usados como novo domínio de conhecimento, caso seja um processo iterativo em que as novas informações são reutilizadas nas iterações seguintes (Frawley, et al., 1991).

2.1.4 Problemas das fontes de dados

Os dados se não forem devidamente tratados, podem produzir conclusões desajustadas da realidade, por isso só devem ser aplicadas técnicas de mineração depois destes serem devidamente analisados. Isto está relacionado com o facto de estes não serem, por vezes ajustados e controlados, serem incompletos e não seguirem constantemente uma estrutura pré-definida. Em particular, estes problemas surgem a partir do momento em que os esquemas de dados do mundo real são dinâmicos, incompletos, ruidosos (possuem informação desnecessária por vezes), e, com um elevado volume. Com isto, surgem algumas preocupações, tais como, se existe informação de interesse para o domínio para ser descoberta, ou se é constituída excessivamente por informações irrelevantes para o mesmo.

Todas fontes de dados partilham de uma característica comum: o conteúdo dos dados encontra-se em constante mudança, o que significa que o conhecimento extraído apenas se irá aplicar até determinado ponto no tempo, já que posteriormente podem ocorrer alterações e com isso negar os possíveis padrões encontrados.

Não só a constante mudança nos dados, mas também o conteúdo irrelevante que estes possam conter para um determinado domínio de conhecimento, pode variar ao longo do tempo. Essa variação poderá ter influência na informação que se extrai tendo em conta o objetivo dessa extração. Caso se esteja a extrair informação acerca da residência dos clientes, não é importante saber se ele tem filhos ou não, mas caso se esteja a analisar a informação de uma base de dados de uma empresa de brinquedos, essa informação já se mostra relevante.

Assim como os dados podem ser considerados relevantes ou não para um determinado fim, existe a necessidade de que todos os atributos se encontrem preenchidos, caso contrário pode estar a influenciar de forma errada a extração do conhecimento. Para isso, devem ser usadas técnicas para determinar o valor desses atributos por forma a diminuir este tipo de ocorrências, ou então, deve definir-se valores por defeito, só que nestes casos devem ser tidos em conta quais os atributos em que se pode e deve fazer, uma vez que nem todos são possíveis de serem tratados.

Também o tipo de dados associados aos atributos pode originar uma variedade de erros, já que podem ser de diversos tipos de dados, e, por exemplo, assumindo a possibilidade de num determinado atributo apenas constarem números, e, no entanto, estarem a surgir caracteres não numéricos, devido ao tipo de dados associado a esse atributo, gerando alguma

incerteza nos mesmos. Não só, mas também a exatidão dos dados, pode causar alguma incerteza, ou seja, a exatidão está associada também ao nível de ruído presente nos mesmos, isto é, por vezes nos dados surgem informações desnecessárias, ou valores fora do contexto para as análises, mas que podem ser a causa de uma classificação não correta.

Por último, mas não menos importante, tem-se os campos em falta. Uma vista errada do esquema de dados pode não marcar determinados dados como corretos, assim sendo, uma vista do esquema de dados deverá conter a totalidade dos atributos úteis que os sistemas possam recorrer durante o processo, já que os atributos são assumidos como sendo diferenciadores nos casos de interesse, e a falha de um campo pode distinguir registos que seriam iguais, sugerindo ações diferentes (Frawley, et al., 1991).

2.2 Problema da Reutilização de Processos de Descoberta de Conhecimento

Nos dias de hoje existem muitas análises que são feitas sobre os dados mas que não podem ser partilhadas, por vezes, entre esquemas semelhantes mas não rigorosamente iguais pertencentes a um mesmo domínio. Tal situação acontece porque os esquemas de dados nem sempre possuem os mesmos atributos, ou o mesmo atributo encontra-se representado do modo diferente, ou até mesmo porque possui atributos que são utilizados e que necessitam de ser descartados. Por vezes, o tratamento da heterogeneidade dos dados leva a que o analista sempre que queira realizar determinada análise sobre novos esquemas do mesmo domínio do anterior, tenha de alterar ou recomeçar o processo de *KDD* para extrair o conhecimento que pretende. Por exemplo, o analista constrói um processo de descoberta de conhecimento ajustado a uma determinada fonte de dados A (ver Tabela 1), entretanto a empresa funde-se com uma outra e pretende extrair o conhecimento que se encontra na fonte de dados B que diz respeito à empresa com a qual se fundiu. O analista neste caso terá de readaptar o processo à nova fonte e com isto irá despender tempo que seria útil para aplicar, por exemplo, outras técnicas de mineração de dados.

Na Tabela 1 está representado o esquema de dados A referente à fonte de dados A, que possui quatro atributos, mas no entanto é necessário derivar um quinto atributo referente ao saldo anterior ao movimento para que as técnicas de mineração a serem aplicadas consigam extrair mais conhecimento. Após este passo de transformação, que se encontra evidenciado na Figura 3, obtém-se um novo esquema de dados, estando ele representado na Tabela 2.

Tabela 1 - Esquema de dados da Fonte de dados A

Esquema de dados A
CardID
Data
Valor
SaldoFinal

Na Figura 3 está a ser adicionado um novo atributo que reutiliza os atributos existentes no esquema A, sendo que para obter o "SaldoAnterior" é o resultado da soma do atributo "Valor" com o "SaldoFinal".

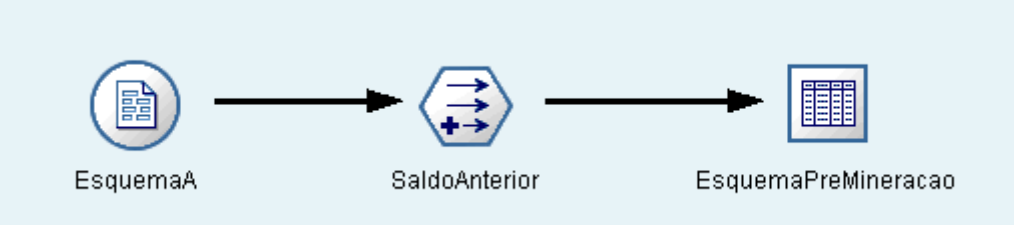

Figura 3 – Exemplo do processo de transformação aplicado a uma fonte de dados (SPSS Modeler)

Tabela 2 - Esquema de dados pré-mineração

Esquema de dados pré-mineração
CardID
Data
Valor
SaldoAnterior
SaldoFinal

Segundo o exemplo dado anteriormente, existindo uma nova fonte de dados com o esquema que se encontra representado na Tabela 3, é necessário extrair junto dessa quais os atributos que possui, e aplicar os componentes adequados para se conseguir aplicar o mesmo processo e obter o mesmo esquema de dados pré-mineração, isto é, sobre o qual serão aplicados os algoritmos de mineração de dados. Como tal o analista teria de criar pelo menos

um componente para calcular o atributo em falta (“SaldoFinal”), isto é, criar o componente e configurá-lo para que seja gerado o atributo, e é neste tipo de configuração que o analista despende tempo, uma vez que no esquema de dados pré-mineração (o esquema de dados que serve de base à fase de mineração de dados) existe esse campo, e como tal, existe a possibilidade de se obter informação que permita inferir o cálculo do atributo em falta sem a intervenção do analista.

Tabela 3 - Exemplo do esquema de dados B

Esquema de dados B
CardID
Data
Valor
SaldoAnterior

Quando se tenta aplicar o mesmo processo sobre uma nova fonte, existe a necessidade de readaptação do processo de descoberta de conhecimento desenvolvido mas, no entanto, é possível afirmar que nem todas as fases precisam de sofrer uma reestruturação/adaptação. Uma vez que o objetivo é aplicar as mesmas técnicas de mineração de dados que tinham sido aplicadas sobre a primeira fonte, logo o que é necessário é refazer as primeiras fases do KDD e aproveitar assim as restantes fases do processo anterior. Para tal, deve-se aproveitar o esquema produzido (esquema de dados pré-mineração) no final da fase de transformação e que deverá ser respeitado sempre que este processo de descoberta de conhecimento seja aplicado a uma nova fonte com um esquema diferente.

O objetivo desta dissertação é criar uma metodologia que permita otimizar as tarefas desenvolvidas pelos analistas, para que estes não despendam demasiado tempo na configuração e na transformação dos dados, rentabilizando esse mesmo tempo na extração de conhecimento ou em outras atividades. Desta forma, esta metodologia irá abranger não só a preparação dos dados, como também o pré-processamento dos mesmos, permitindo que possam ser executados os algoritmos de mineração de dados sobre os mesmos, recorrendo apenas a ajustes, se necessário, quando aplicado sobre um outro esquema.

2.3 CRISP-DM

O "Cross Industry Standard Process for Data Mining" (CRISP-DM) é um projeto destinado a colmatar as necessidades da mineração de dados com o objetivo de prevenir falhas. Note-se que no contexto desta metodologia, mineração de dados é utilizado com o sentido lato de processo de descoberta de conhecimento.

"A mineração de dados necessita de uma abordagem padrão que irá ajudar a traduzir problemas de negócio nas tarefas de mineração de dados, sugerindo transformações de dados apropriados e técnicas de mineração de dados, e fornece significado para avaliar a eficácia dos resultados e documentando a experiência" (traduzido de (Wirth, et al., 2016)).

A mineração de dados é um processo complexo que necessita de várias ferramentas e recorre a diferentes pessoas, sendo que o sucesso dos projetos está dependente da adequada combinação entre as ferramentas e as qualidades dos analistas.

O CRISP-DM tem como principal objetivo diminuir os custos, tornar os projetos de mineração de dados mais confiáveis, mais repetitivos, mais maneáveis e mais rápido (Wirth, et al., 2016).

A aceitação de um modelo de processo comum traz muitos benefícios, sendo que esse modelo pode ser usado como ponto de referência comum para discussão acerca da mineração de dados, permitindo uma maior compreensão dos problemas cruciais da mineração de dados, criando deste modo uma impressão de que o processo se encontra estável (Wirth, et al., 2016).

2.3.1 Fases do CRISP-DM

O CRISP-DM pode ser interpretado como sendo um ciclo de fases em torno do processo de descoberta de conhecimento, sendo que a sequência dessas fases não é rigorosa, existindo a possibilidade em algumas fases de voltar a fases anteriores, como indicam as setas na Figura 4.

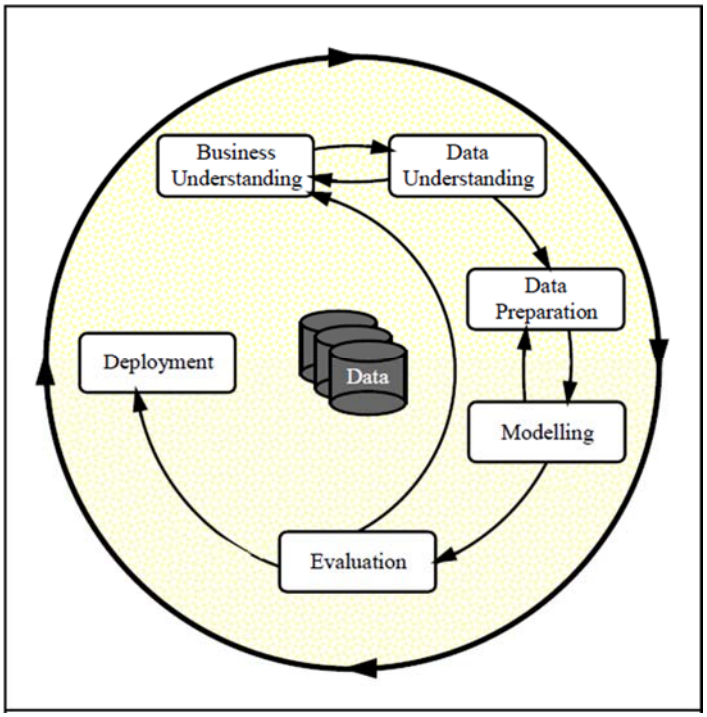

Figura 4 – Fases do CRISP-DM (Wirth, et al., 2016)

Deste modo as fases pertencentes ao CRISP-DM são: a compreensão do negócio, a compreensão dos dados, a preparação dos dados, a modelação, a avaliação e a publicação dos resultados.

Assim como referido anteriormente, as fases não tem uma sequência definida sendo que o projeto deste modo vai evoluindo com as fases anteriores, ou seja com a experiência. Um exemplo disso é a fase de preparação dos dados que pode ser melhorada através da experiência e com base na fase de modelação, em que pode necessitar de algum ajuste nos dados, ou a categorização de algum atributo (Wirth, et al., 2016).

Breve descrição das fases:

- Compreensão do negócio – Esta é a fase inicial focada na compreensão dos objetivos do projeto e dos requisitos, convertendo o conhecimento adquirido nisso na definição de um problema de mineração de dados.
- Compreensão dos dados – Inicia-se com o processamento da coleção de dados inicial e procede com as atividades por forma a identificar problemas nos dados, e subconjuntos de dados relevantes.

- Preparação dos dados – Engloba todas as atividades para contruir o conjunto de dados final, podendo ser executada múltiplas vezes. Esta é uma fase que engloba a limpeza dos dados, seleção de atributos, criação de novos atributos e a transformação dos dados.
- Modelação – consiste em recorrer a técnicas de mineração de dados, sendo que por vezes, para o mesmo problema existe mais do que uma técnica que pode ser aplicada.
- Avaliação – é a fase em que os modelos já se encontram construídos com base nas definições das fases anteriores, mas que são submetidos a avaliações. Nesta fase os modelos e as técnicas são avaliadas, por forma a verificar se existe algum problema com o modelo construído e se respeita as condições de negócio estabelecidas.
- Publicação – publicação/disponibilização do modelo em que expõe o conhecimento extraído dos dados, sendo que este modelo pode necessitar de ser refinado. Isto irá depender dos requisitos o que pode ser tão simples como gerar relatórios, ou então implementar uma solução que permita a repetição da fase da modelação.

2.4 PMML

De um modo geral, o PMML é uma descrição de modelos com base em XML que permite a portabilidade do modelo gerado entre diferentes plataformas.

"O PMML (*Predictive Model Markup Language*) é descrito como sendo uma linguagem de marcação baseada em XML para trabalhar com modelos preditivos produzidos por sistemas de mineração de dados. Esta linguagem pode ser usada para definir modelos preditivos ou conjunto de modelos preditivos. […] Este revelou-se útil para aplicações que requerem aprendizagem particionada, aprendizagem de conjuntos e aprendizagem distribuída." (traduzido de (Grossman, et al., 2002)).

A cultura da mineração preditiva[7] está preocupada com a exploração de algoritmos de mineração de dados para produção de modelos preditivos automaticamente (Grossman, 1998), por sua vez, a cultura de descoberta de conhecimento está preocupada com a exploração de algoritmos de mineração de dados para extrair conhecimento válido, novo e útil (Fayyad, 1996).

[7] Mineração preditiva – traduzido do inglês Predictive Mining

Os modelos descritos pelo PMML são compostos por: cabeçalho, esquema de dados, esquema de mineração de dados, esquema do modelo preditivo, definições para os modelos preditivos, definições para os conjuntos de modelos, regras para selecionar e combinar modelos e conjuntos de modelos, e ainda regras para as exceções que forem encontradas. Mas, destas componentes apenas as definições dos modelos preditivos são obrigatórios, assim como o esquema para o modelo preditivo, que pode recorrer a pelo menos um dos esquemas referidos anteriormente.

O PMML, como já foi dito anteriormente, é uma linguagem de anotação, sendo que se encontra estruturado conforme o exemplo que se encontra a seguir, em que é possível consultar a estrutura do esquema de dados, o esquema do modelo e a descrição do algoritmo utilizado no cálculo do modelo.

```
<pmml>
<data-schema>
<attribute-descriptor attribute-number='1' attribute-name='card number'
use-as='exclude' data-type='string'>
<attribute-descriptor attribute-number='2' attribute-name='timestamp'
use-as='exclude' data-type='string'>
<attribute-descriptor attribute-number='3' attribute-name='dollar amount'
use-as='continuous' data-type='real'>
<attribute-descriptor attribute-number='4' attribute-name='issuer' useas='
category' data-type='integer'>
</data-schema>
<model-schema>
<attribute-descriptor attribute-number='1' attribute-name='number of
transactions past hour' corresponds-to='data-attribute 12' datatype='
integer'>
<attribute-descriptor attribute-number='2' attribute-name='number of
transactions past two hours' corresponds-to='data-attribute 15' datatype='
integer'>
</model-schema>
<cart-model type='binary-classification' attribute-predicted='Fraud
Indicator' number-nodes='13' depth='3'>
<cart-node node-number='0' attribute-number='model-attribute 1'
cut-value='3' left-child='1' right-child='6'>
<cart-node node-number='1' attribute-number='model-attribute 3'
cut-value='5' left-child='2' right-child='3'>
</cart-model>
</pmml>
```

Código 2 – Conteúdo de um ficheiro PMML exemplo usando CART

2.5 Ferramentas de Descoberta de Conhecimento

As ferramentas de descoberta de conhecimento são utlizadas por analistas que, na maioria das vezes, criam processos de descoberta de conhecimento baseados em cada base de dados. Normalmente estas ferramentas são conhecidas por possuírem um ambiente gráfico em que os seus utilizadores definem os componentes que querem usar, assim como o fluxo que deverá ser executado.

Nesta secção são abordadas as ferramentas que se encontram nos topos das listas de ferramentas destinadas ao processo de descoberta de conhecimento. No topo da lista das ferramentas com uma licença não gratuita está o *IBM SPSS Modeler* (designado anteriormente de *Clementine*), no topo da lista das gratuitas está o WEKA (listas consultadas em abril de 2016).

2.5.1 IBM SPSS Modeler

O *IBM SPSS Modeler* (IBM ;, 2016) (ou *Clementine* inicialmente) é considerado dos melhores *software*s para realizar operações de mineração de dados sobre um conjunto de dados já transformados anteriormente. Apresenta-se como sendo uma plataforma de análise preditiva que foi concebida para fazer chegar a inteligência preditiva às decisões tomadas por indivíduos, grupos, sistemas e pela própria empresa. Fornece vários algoritmos avançados e técnicas que incluem análise de texto, análise de entidades, gestão de decisões e otimização, para o ajudar a selecionar as ações que terão melhores resultados. Disponível em várias edições, incluindo uma versão baseada na nuvem[8].

Disponibiliza uma interface gráfica em que o utilizador apenas tem de configurar os componentes que achar necessários para as operações que pretender realizar, criando uma *stream* com o aspeto idêntico à que está representada na Figura 5.

[8] Nuvem – traduzido do termo em inglês Cloud

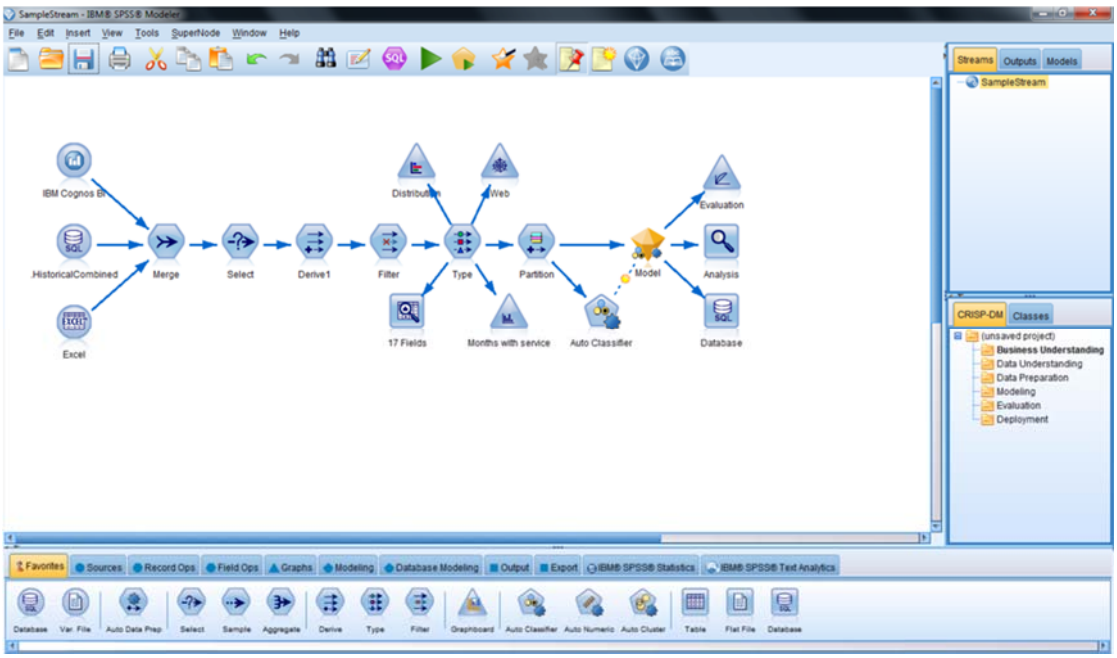

Figura 5 – Exemplo de um processo de descoberta de conhecimento no *IBM SPSS Modeler* (IBM;, 2016)

Segundo dados estatísticos, este é uma das ferramentas de descoberta de conhecimento mais utilizada a nível mundial. Disponibiliza uma quantidade considerável de algoritmos que permitem ao utilizador extrair os dados conforme o seu objetivo (IBM ;, 2016).

O *IBM SPSS Modeler* é um *software* que engloba todas as fases de um processo de descoberta de conhecimento, e como tal, um dos motivos pelo qual é dos mais requisitados. Permite a visualização dos resultados em diferentes formatos, para que o utilizador, ou seja, o analista, consiga identificar padrões através da visualização, ou até mesmo identificar quais as regras que estão a ser aplicadas nos algoritmos de classificação.

2.5.1.1 CLEM - IBM® SPSS@ Modeler scripting language

O *software IBM SPSS Modeler* permite a inclusão de excertos de código através da sua própria linguagem. O *Control Language for Expression Manipulation (*CLEM) permite a manipulação e execução de nós, incluindo a criação dos mesmos, ou seja, é possível manipular um nó sem recorrer à interface do *software*, e usando uma linguagem própria do mesmo.

"O CLEM é uma linguagem poderosa para analisar e manipular os dados que fluem ao longo das streams do *IBM SPSS Modeler*. Os analistas usam o CLEM extensivamente nas operações da stream para realizar tarefas tão simples como a obtenção do lucro a partir de dados de custo e receitas, ou tarefas mais complexas como transformar dados web num

conjunto de campos e registos com informação útil" (traduzido de (IBM Knowledge Center, 2012)) (ver Código 3).

O CLEM é usado dentro do SPSS Modeler para:

- Comparar e avaliar as condições em campos dos registos;
- Derivar valores para novos campos;
- Derivar novos valores para campos existentes;
- Inserir dados a partir de registos noutros relatórios (IBM Knowledge Center, 2012).

Por vezes, esta linguagem também é usada dentro dos componentes do *IBM SPSS Modeler* para criar, por exemplo, novos campos e conseguir com recurso ao CLEM formar a expressão necessária (IBM Knowledge Center, 2012).

```
open stream "$CLEO_DEMOS/streams/druglearn.str"
execute :c50node
save model Drug as rule.gm
clear generated palette
open stream "$CLEO_DEMOS/streams/drugplot.str"
load model rule.gm
disconnect :plotnode
insert model Drug connected between :derive and :plot
set :plotnode.color_field = '$C-Drug'
execute :plotnode
```

Código 3 – Exemplo de um script CLEM para gerar um modelo e carregá-lo

2.5.2 KNIME

KNIME é o acrónimo de *Konstanz Information Miner*, e é uma fonte aberta de dados analíticos, elaboração de relatórios e uma plataforma de integração.

Este surgiu em 2004 na Universidade de Kostanz, através de uma equipa de programadores de uma empresa de Silicon Valley especializada em aplicações farmacêuticas e que iniciou o trabalho numa plataforma aberta. Esta tinha como objetivo ser usada como ferramenta de colaboração e de investigação, até que este produto começou a processar e a integrar grandes quantidades de dados diversos, e os programadores aderiram e criaram uma

plataforma que abrangesse o carregamento de dados, transformação, análise e exploração visual dos modelos. E em 2006 é lançada a primeira versão do KNIME (Knime, 2016).

Atualmente o KNIME suporta uma variedade de áreas desde investigação cientifica, serviços financeiros, editores, retalhistas, empresas de consultoria, governos e investigação.

KNIME está desenvolvido em Java, é baseado no Eclipse e é uma ferramenta gratuita. Este disponibiliza vários módulos, sendo que no âmbito desta dissertação apenas será focado o módulo KNIME Analytics Platform, que possui o aspeto que se encontra na Figura 6.

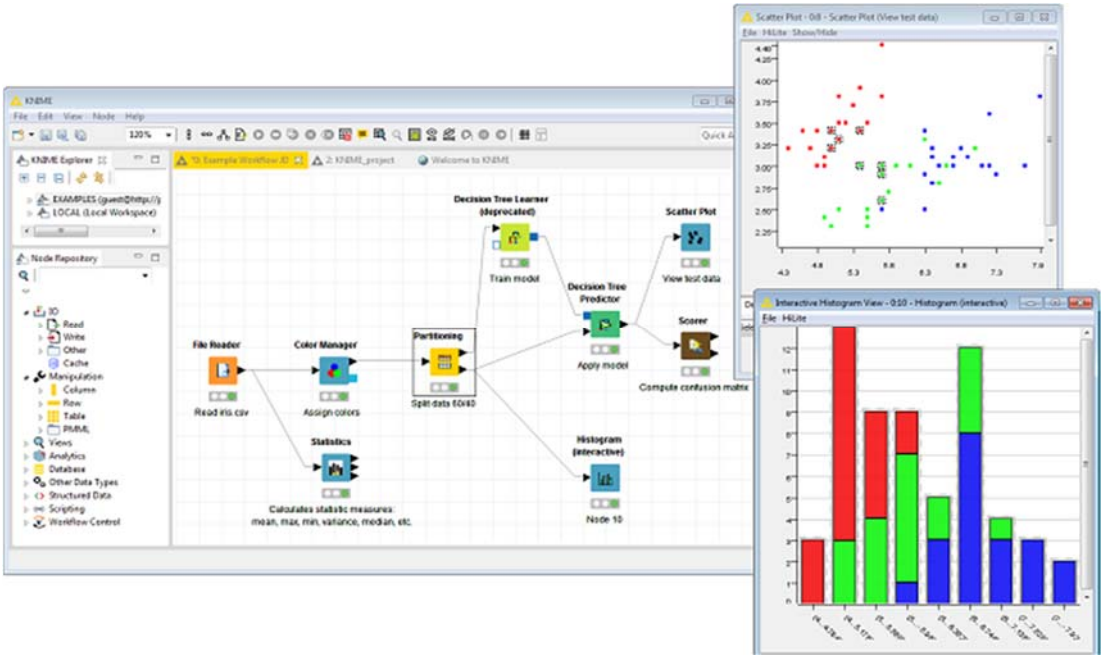

Figura 6 – Possíveis análises que se conseguem extrair com recurso ao KNIME (KNIME, 2016)

O módulo KNIME Analytics Platform é a principal solução aberta para a inovação com base em dados, ajudando a descobrir o potencial escondido nos dados, ou para prever novos futuros.

Esta é uma ferramenta que já conta com 18 versões desde 2008 e com milhares de utilizadores. Permite estender as suas capacidades, adicionando módulos comerciais para vários fins.

Ao nível dos dados, este permite que sejam combinadas várias fontes, desde ficheiros de texto, a base de dados, documentos, imagens, entre outros, disponibilizando componentes que permitem a adição de código fonte em *Python, R, SQL* e *Java* para uma melhor extração dos dados, possuindo uma interface visual simples de compreender (KNIME, 2016).

2.5.3 WEKA

Waikato Environment for Knowledge Analysis (WEKA) surgiu para tentar resolver os problemas práticos presentes na indústria da Nova Zelândia aplicando técnicas de descoberta de conhecimento, e sendo um dos seus objetivos a partilha de novos algoritmos de mineração de dados por todo o Mundo.

Com o WEKA, um especialista numa área específica é capaz de usar o KDD para obter conhecimento a partir das bases de dados que são, por vezes, extensas para serem analisadas à mão. Os seus utilizadores são investigadores de KDD e cientistas industriais, mas este também é usado no ensino (WEKA;, 2016).

WEKA é uma coleção de algoritmos de aprendizagem máquina para tarefas de descoberta de conhecimento. Esses algoritmos podem ser tanto aplicados a um conjunto de dados como invocados a partir de um desenvolvimento em Java. WEKA contém ferramentas para pré-processamento, classificação, agrupamentos, regressão, regras de associação, e visualização de dados. Este é também adequado para o desenvolvimento de novos sistemas de aprendizagem máquina (Weka, 2016).

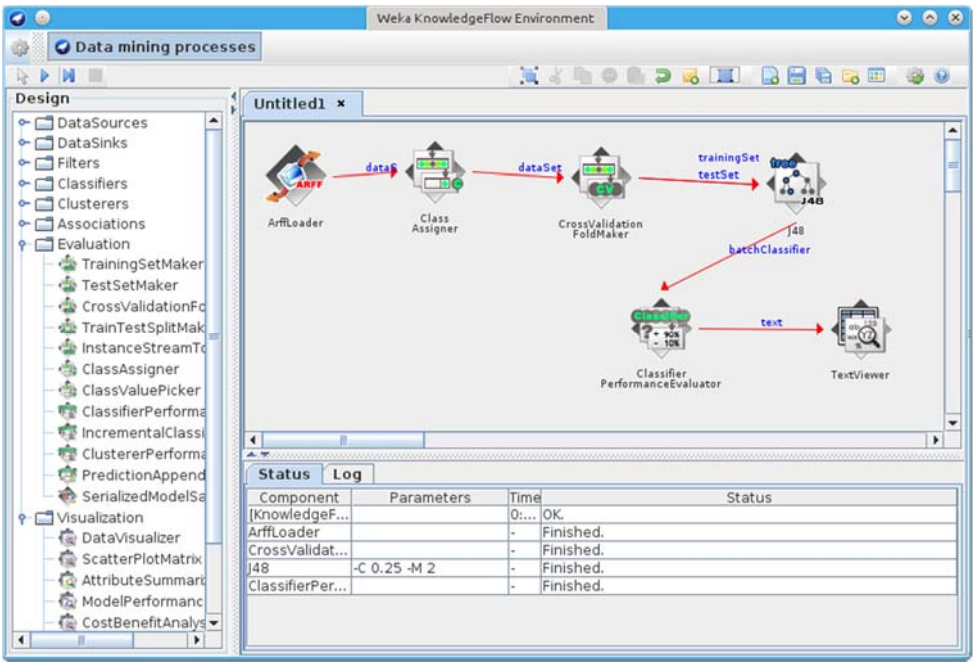

Figura 7 – WEKA KnowledgeFlow (WEKA, 2016)

2.5.4 Comparação Ferramentas de Descoberta de Conhecimento

Na Tabela 4 está presente uma comparação das ferramentas descritas anteriormente. Nesta tabela cada uma das ferramentas é classificada conforme a sua facilidade de utilização, o seu preço, a quantidade de algoritmos que disponibiliza, e a facilidade de manipular dados.

Tabela 4 – Comparação das ferramentas de Descoberta de Conhecimento

	Facilidade de utilização	**Preço**	**Algoritmos de análise**	**Manipulação de dados**
IBM SPSS Modeler	Boa	Elevado	Bastante completa	Excelente
KNIME	Boa	Grátis	Quantidade aceitável	Fraca
WEKA	Boa	Grátis	Quantidade aceitável	Média

Como resultado desta comparação, o *IBM SPSS Modeler* é uma ferramenta bastante completa, sendo que a sua única desvantagem é o preço. No entanto, entre as restantes duas ferramentas o que se pode concluir é que são semelhantes, mas o WEKA permite manipular mais facilmente os dados em comparação direta com o KNIME.

2.6 OntoDataClean

Uma vez que as ontologias podem fazer parte da metodologia a ser seguida, na literatura encontrou-se este projeto que usa as ontologias para mapear os atributos com o mesmo significado, mas com designações distintas.

OntoDataClean é um projeto desenvolvido com base em ontologias que se foca na fase de pré-processamento do KDD. Esta é uma fase que exige um maior gasto de tempo, uma vez que é nela que se começam a delinear os campos que interessam manter, ou descartar, ou identificar os que necessitam de uma transformação (Perez-Rey, et al., 2006).

A mineração de dados é o passo principal do KDD, no entanto as fases anteriores são cruciais para assegurar a integração, a compilação e a qualidade dos dados, isto é, as fases de seleção, de pré-processamento e de transformação (Perez-Rey, et al., 2006).

A integração, o pré-processamento e a transformação de registos armazenados em diferentes bases de dados é o foco do sistema, recorrendo a ontologias, em que o objetivo foi

criar uma ferramenta intuitiva onde os analistas modelam os dados e armazenam a informação necessária para transformar tais dados.

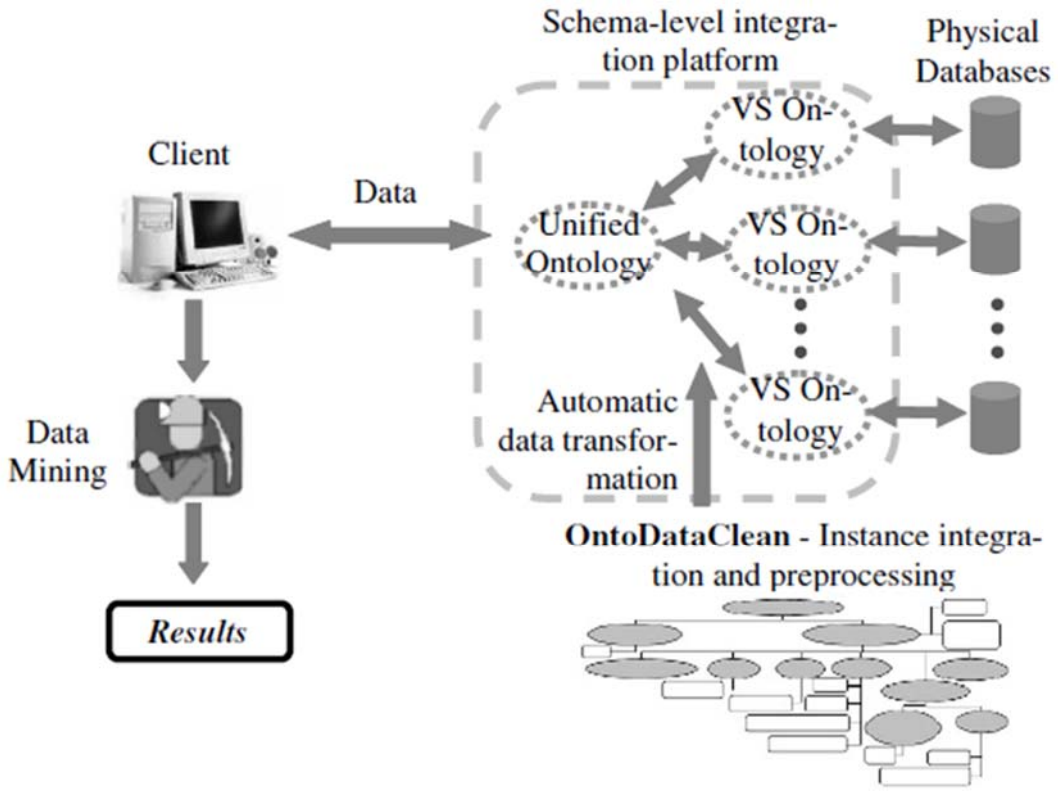

Figura 8 – Abordagem do KDD baseado em ontologias (Perez-Rey, et al., 2006)

Como é possível verificar na Figura 8, esta abordagem centra-se no uso de múltiplas ontologias através de uma única ontologia que fica exposta para as comunicações externas. Nesta abordagem, as ontologias são esquemas de dados virtuais em que cada uma representa uma fonte de dados diferente, não armazenando dados, sendo que os pedidos que são efetuados sobre os esquemas de entidades relacionadas podem ser efetuados sobre as ontologias, uma vez que estas também suportam esses. Estas ontologias representam apenas a fase de pré-processamento do KDD, sendo que a fase de transformação se encontra oculta no ponto de ligação entre a ontologia disponibilizada pelo processo e cada uma das restantes.

Este é um projeto direcionado para dados biomédicos, dados esses que podem ser de diversos tipos, como textuais ou dados quantitativos, em que em ambos os casos o projeto consegue lidar corretamente com cada um (Perez-Rey, et al., 2006).

Desta forma, o OntoDataClean é um projeto que permite o mapeamento de múltiplas fontes de dados através de um mapeamento direto entre a fonte de dados e a ontologia. Os dados são transformados automaticamente através de uma ontologia que se encontra

disponível apenas para a fase de mineração, sendo este o ponto de ligação entre a fase de transformação e de mineração de dados.

2.7 Correspondência de esquema

A correspondência de esquemas também se apresentou como uma possibilidade para vir a ser usada na metodologia a ser desenvolvida, sendo que poderia ser importante para o mapeamento entre os diversos esquemas que foram referidos no exemplo apresentado anteriormente.

"Correspondência de esquema [9] é o problema de gerar correspondências entre elementos de dois esquemas. […] A correspondência é uma relação entre um ou mais elementos de um esquema e um ou mais elementos de outro (traduzido de (Bernstein, et al., 2011))", entenda-se por esquema uma estrutura formal que representa um artefacto de engenharia, que pode ser visto como um esquema em SQL, em XML (XSD é a definição da estrutura, das regras que o XML deverá obedecer), ou um diagrama de entidades. "A correspondência é uma relação entre um ou mais elementos de um esquema e um ou mais elementos de um outro esquema" (traduzido de (Bernstein, et al., 2001)). Esta correspondência tem semântica se esta restringe as instâncias dos elementos do esquema relacionado.

2.7.1 Técnicas de correspondência de esquema

Na correspondência de esquema existem técnicas a serem adotadas para que se faça corresponder os diferentes elementos em cada um dos esquemas.

As técnicas apresentadas a seguir foram identificadas no ano 2001 e são descritas como sendo as primeiras técnicas de correspondência de esquemas que surgiram, sendo elas (Bernstein, et al., 2001):

- Correspondência linguística é baseada nos nomes ou descrições dos elementos, recorrendo a tokenização, correspondência a cadeia de caracteres ou subcadeias de caracteres, e técnicas de recuperação de informação;

[9] Correspondência de esquema – traduzido do inglês *Schema Matching*

- Uso de informação auxiliar baseada em acrónimos, dicionários e listas de incompatibilidades;
- Correspondência baseada na instância refere-se a elementos do esquema que são considerados como similares, caso as suas instâncias o sejam, isto é, se dados estatísticos, meta dados ou classificadores treinados assim o indicarem;
- Correspondência baseada na estrutura refere-se a elementos do esquema que são similares se surgem em grupos similares estruturados, se tem relações similares, ou tem relações com elementos similares;
- Correspondência baseada em restrições refere-se aos tipos de dados, ao intervalo de valores, singularidade, valores nulos e chaves estrangeiras;
- Correspondência baseada em regras refere-se à correspondência das regras que são expressas numa lógica de primeira ordem;
- Correspondência híbrida é uma combinação de algoritmos, sendo eles de correspondência linguística, correspondência baseada na estrutura, restrições ou contexto.

Desde 2001, foram desenvolvidas novas técnicas que recorrem a novos tipos de informação, tais como:

- Correspondências de gráficos que se baseia na comparação de relações entre elementos nos esquemas de gráficos;
- Correspondência baseada na utilização que se baseia na análise aos registos das questões nas bases de dados para dicas sobre como utilizadores relacionam esquemas (Elmeleegy, 2008). Caminhos de taxonomia podem ser correspondidos por encontrarem páginas web que representam os caminhos e então analisando os registos das palavras-chave das questões para determinar se as páginas são acedidas através de questões similares;
- Similaridade do conteúdo do documento onde instâncias de um elemento esquema são agrupadas num documento que é então correspondido com outro, tais documentos baseados na medida de recuperação de informação TF-IDF[10] (Li, 2009);
- Similaridade da ligação do documento onde conceitos em duas ontologias são considerados como similares se as entidades que se referem a esses conceitos são semelhantes.

[10] TF-IDF – Frequência do termo inverso da frequência nos documentos;

Para além destas técnicas apresentadas anteriormente, têm sido propostas estratégias para combinar, de forma flexível, múltiplos algoritmos de correspondência, principalmente para comparar esquemas grandes, tais como:

- Ordem de trabalho[11] – estratégias para independentemente ou sequencialmente executar algoritmos de correspondência e para combinar os seus resultados (Bernstein, et al., 2004);
- Onde são dadas tarefas de correspondência dos domínios de tarefas de correspondência, um ajustador seleciona os componentes de corresponder para serem combinados e/ou atribuir valores aos vários parâmetros que afeta como os resultados correspondentes do componente são combinados (Li, 2009);
- A busca precoce de espaço de poda[12] – onde um combinador rápido é usado para eliminar correspondências indesejadas a partir da consideração para que um conjunto pequeno de elementos maneáveis possa ser correspondido usando mais técnicas dispendiosas e precisas (Quick ontology matching, 2004);
- Correspondência baseada na partição baseia-se na redução do espaço de correspondências possíveis, os esquemas de entrada são particionados seguidos por correspondência de partição sábia (Do, et al., 2007);
- Correspondência paralela onde os diferentes passos de algoritmo de correspondência são executados em paralelo, ou diferentes partições de esquemas são combinadas em paralelo (Gross, et al., 2010);
- Otimizações para esquemas de dados grandes, tais como otimizações de combinação de cadeia de caracteres, pré-recolhendo antecessores e "filhos" de casa elemento para evitar a travessia repetida, e usando matrizes de similaridade de espaço eficiente (Bernstein, et al., 2004).

Abordagens que são propostas quando vários esquemas num domínio são combinados coletivamente. Como por exemplo:

- Combinação baseada na reutilização é quando combinações entre fragmentos de esquemas são colhidos a partir de mapeamentos validados e usados como informação auxiliar para ajudar tarefas futuras no mesmo domínio (Madhavan, et al., 2005);

[11] Ordem de trabalho – traduzido do inglês Workflow;
[12] A busca precoce de espaço de poda – traduzido do inglês Early search space pruning.

- Combinação holística diz respeito a quando um simples esquema mediado está construído para um domínio pelos elementos de alinhamento de um grande conjunto de esquemas, tais como formulários web cobrindo um domínio particular (He, et al., 2003);

Entretanto foram surgindo estratégias para incorporar a interação e o feedback do utilizador no processo de correspondência, tais como: Combinações incrementais, combinações Top-k, e um suporte gráfico para inspecionar e corrigir as correspondências calculadas pelos sistemas.

Por fim, foram adicionados algoritmos para etiquetar as correspondências conforme digam respeito a uma correspondência de semântica, ou condicional (Gal, 2011) (Bernstein, et al., 2011).

3 Solução proposta

Perante o problema apresentado no Contexto e Estado da Arte, existem algumas considerações a serem tidas em conta, assim como, possíveis tecnologias a serem usadas na solução do problema levantado.

No KDD são consideradas cinco fases, que já foram referidas na Secção 2.1.2 - “Fases do Processo de Descoberta de Conhecimento em Bases de Dados”. Perante estas fases mostra-se viável identificar que os possíveis ajustes sejam ao nível das três primeiras fases, e algumas na quarta fase (mineração dos dados), isto quando o analista apenas pretende aplicar o mesmo processo de descoberta de conhecimento a outra fonte de dados do mesmo domínio. Isto é, o analista precisa de ajustar o esquema de dados que se encontra associado à nova fonte ao esquema pré-mineração da fonte inicial, para que possa reutilizar o processo que se encontra implementado, criando, ajustando e removendo os atributos que se mostrem necessários.

Deste modo, a reutilização do KDD foca-se nas três primeiras fases, sendo que as restantes ficarão estáticas, ou apenas sofrendo pequenos ajustes, caso seja necessário. Deste modo, é possível dividir o processo de KDD em dois subprocessos tal como se encontra na Figura 9. O primeiro que é destinado a uma preparação e transformação dos dados, de modo a que estes respeitem um determinado esquema de dados que fica pré-estabelecido aquando da definição inicial do processo de descoberta de conhecimento. O segundo subprocesso apenas é criado uma vez, e este não irá depender da fonte de dados que estiver a ser usada, isto é, mantém-se igual, qualquer que seja a fonte de dados que esteja a ser tratada dentro do mesmo domínio.

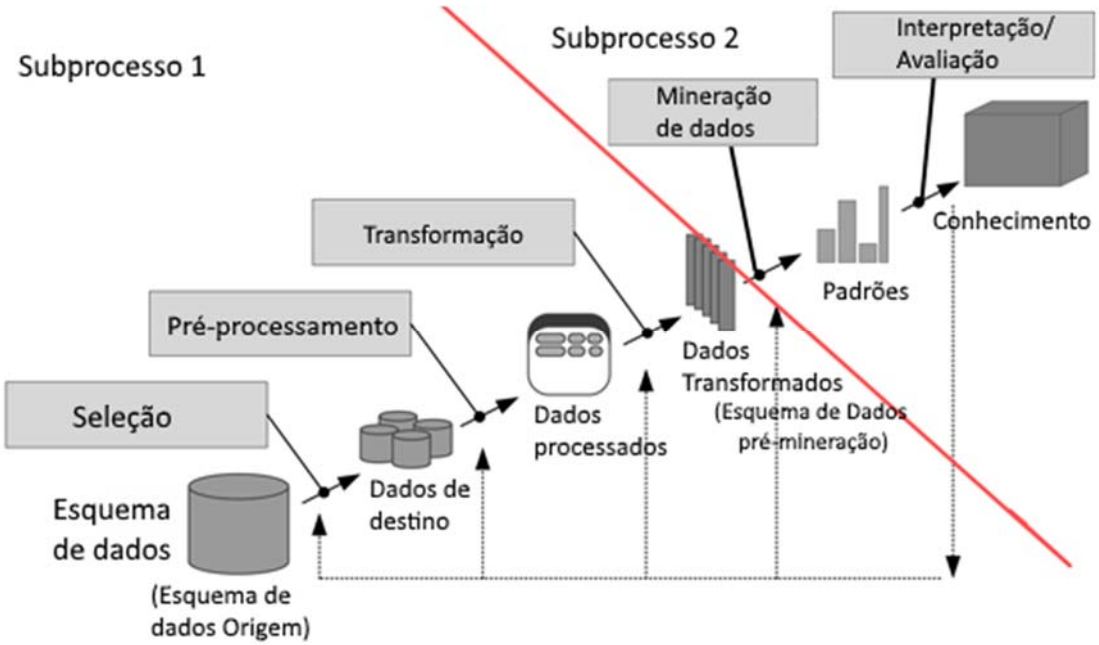

Figura 9 – Divisão do processo de descoberta de conhecimento

O primeiro subprocesso (lado esquerdo da linha vermelha da Figura 9) contém fases que são muito específicas, ou direcionadas para o esquema de dados que está a ser tratado. É sobre este subprocesso que está o foco da solução, uma vez que é nesta parte do processo que se encontra todo o conteúdo que está dependente de cada esquema. Por sua vez, o segundo subprocesso (lado direito da Figura 9), como referido anteriormente, mantém-se constante independentemente da fonte que está a ser manipulada, sendo que para este subprocesso apenas tem de ser garantido de que o resultado do primeiro é o esquema de dados esperado por este (doravante, designado por esquema pré-mineração).

Desta forma, a solução irá centrar-se na transformação do novo esquema de dados selecionado no esquema pré-mineração. Para tal, chegou a ser investigada a inclusão do PMML na solução, mas devido à formulação do problema, este não correspondia ao pretendido em virtude do facto de não permitir fazer corresponder o esquema importado e posteriormente transformado no esquema pretendido pelo segundo subprocesso. Este apenas permite a definição do modelo a ser gerado após aplicação dos algoritmos de mineração dos dados, o que não se aplica ao pretendido uma vez que esses mesmos algoritmos apenas são aplicados no segundo subprocesso.

3.1 Desenho da solução

A solução proposta consiste no desenvolvimento de uma metodologia capaz de identificar e mapear atributos de um esquema de dados com um que é predefinido inicialmente, sendo que, para tal, é usada a estratégia de correspondência semântica para relacionar atributos iguais com designações iguais e aplicar as devidas transformações. Fazendo com que os dados correspondam ao esquema pré-mineração esperado pelo segundo subprocesso.

Desta forma, a abordagem passa por implementar uma solução de correspondência de esquemas em que o objetivo inicial é: transformar o esquema sobre o qual se pretende realizar operações de mineração de dados. Essa transformação inicial consiste no mapeamento dos dados para o esquema pré-mineração (ver Figura 10) identificando quais os atributos que irão ser transformados.

Tal como já foi referido, o problema reside nessas transformações da fonte de dados num esquema de dados já produzido anteriormente. Tal como é possível visualizar na Figura 10, o objetivo desta metodologia é transformar a Fonte de dados B no esquema pré-mineração, que foi o resultado das transformações manuais sobre a Fonte de dados A.

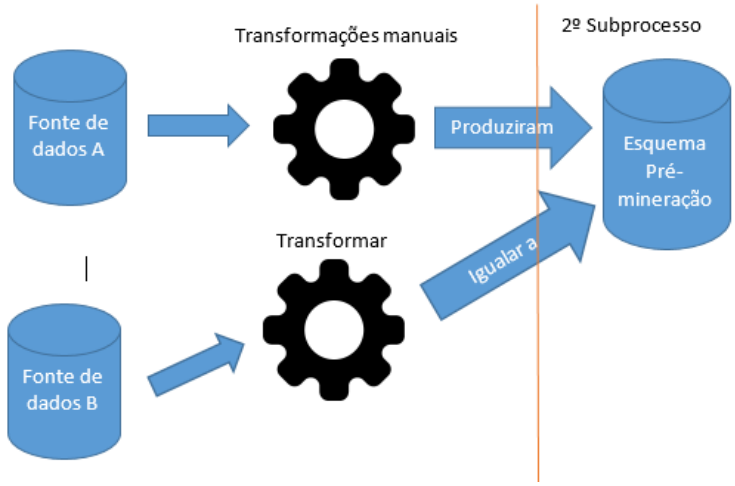

Figura 10 - Objetivo da metodologia

Para que a transformação referida anteriormente tenha sucesso, são adotadas técnicas de pré-processamento pré-definidas que irão permitir agilizar o processo e deste modo criar uma metodologia o mais autónoma possível. Esta será capaz de receber apenas um novo esquema de entrada e aplicar sobre este um mapeamento com base na comparação com a fonte de dados inicial (fonte A) e com o esquema pré-mineração. Após a identificação

das diferenças existentes entre os esquemas, o objetivo é utilizar o esquema B (relativo à fonte B) e aplicar as transformações conforme essas diferenças identificadas, devolvendo um novo esquema de dados que estará em concordância com o esquema desenhado para o segundo subprocesso (esquema pré-mineração).

Na Figura 11 estão evidenciados os principais passos da metodologia criada, tendo em conta que irão englobar 3 fases do processo de descoberta de conhecimento, por forma a torná-las mais independentes da interação humana.

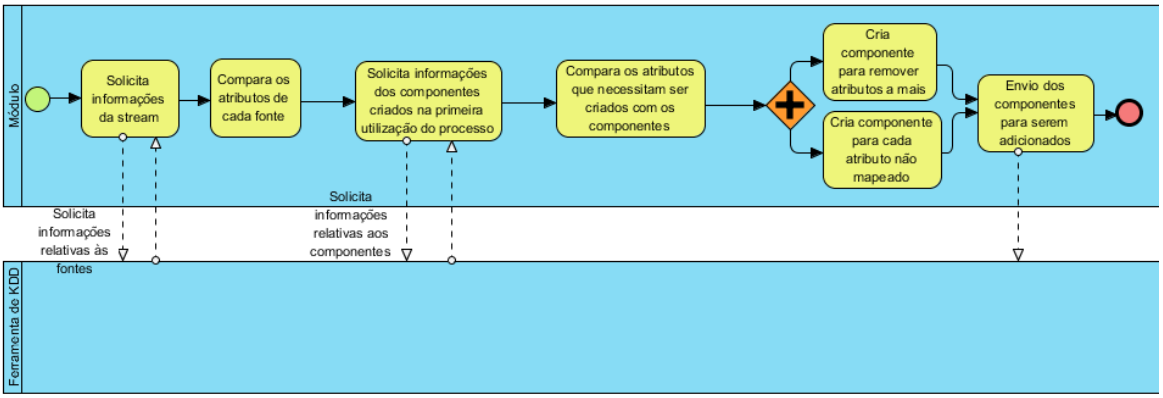

Figura 11 - BPMN da metodologia desenhada

A metodologia desenhada inicia-se com uma leitura da *stream* indicada, pedindo informações acerca das fontes de dados e do esquema de dados pré-mineração aguardado pelo segundo subprocesso. De seguida, passa a avaliar os atributos, indicando as diferenças existentes entre os esquemas, tanto da nova fonte em relação ao esquema pré-mineração (ver Figura 12) como em relação à fonte A (ver Figura 13).

A primeira comparação a ser efetuada é entre o esquema da nova fonte e o esquema pré-mineração, identificando os atributos que necessitam de ser adicionados. No exemplo da Figura 12 é possível verificar que falta o atributo “SaldoFinal” ao esquema B, logo é identificado como sendo um novo atributo a ser adicionado.

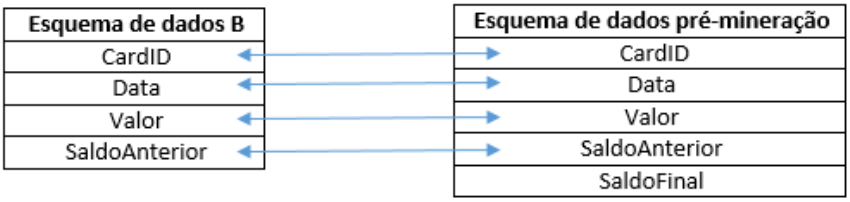

Figura 12 - Campos do esquema de dados B comparados com o esquema de dados pré-mineração

A comparação seguinte é entre a Fonte de dados A e a B, identificando as diferenças em termos de atributos. Na Figura 13 encontram-se ambas as fontes, onde é possível verificar que na fonte de dados A existe um atributo que não existe na B, e que corresponde ao campo já marcado anteriormente como sendo para ser adicionado, sendo que todos os restantes atributos possuem um correspondente.

Figura 13 - Correspondência entre os campos do esquema de dados A e os campos do esquema de dados B

Após essa comparação de atributos, são requisitadas mais informações à ferramenta de KDD acerca dos componentes que foram adicionados para tratar dos atributos da fonte inicial (no exemplo a Fonte de dados A). Após a obtenção dessas informações recorre-se a esses mesmos componentes para auxiliar o processo de criação de atributos em falta na fonte que está a ser manipulada (Fonte de dados B), sendo que a seguir, e por último, se criam os componentes para remover os atributos identificados que não obtiveram correspondência com nenhum do esquema esperado, enviando estes novos componentes para a ferramenta, para que sejam adicionados à *stream*.

Finalmente, depois de se obter o esquema de dados pré-mineração, este é disponibilizado para o segundo subprocesso da descoberta de conhecimento, definido no início deste capítulo, que irá executar as restantes fases do processo de KDD utilizando agora os novos dados já transformados e ajustados ao que era esperado.

3.1.1 Caso de exemplo

Usando o exemplo que consta na Tabela 3, é possível verificar que tanto a fonte de dados A como a fonte de dados B não possuem os mesmos atributos, apesar de possuírem o mesmo número de atributos. No entanto, o esquema pré-mineração foi derivado a partir do esquema da fonte de dados A e das transformações aplicadas a este. Isto é, o analista adicionou o "SaldoAnterior" à fonte A e desta forma definiu que o esquema pré-mineração é constituído pelos campos da fonte de dados A e pelo novo campo adicionado posteriormente, resultando nos 5 campos finais como (

Tabela 2). Com a Fonte de dados B o pretendido é que seja identificado a falta de um campo e que a adição desse mesmo campo seja derivada das operações que foram aplicadas à Fonte de dados A. É a partir dessas operações que é possível adicionar os campos em falta no esquema B.

Na fonte de dados B foi identificado que faltava um campo, o "SaldoFinal", e perante isto, se fosse o processo manual o analista teria de criar este campo, aplicando as operações considerados necessários. No entanto, e seguindo a metodologia desenhada, existem três abordagens para se adicionar um atributo que se encontra em falta:

1. Existe na "fonte de dados A" e é possível determiná-lo com base nalguma operação que exista durante o processo de transformação dessa fonte, isto é, resolver essa operação em ordem ao atributo pretendido;
2. Não existe na "fonte de dados A", no entanto, a operação de adição do campo está contida no processo e não envolve nenhum atributo que não faça parte do esquema B, então é possível aplicar essa mesma operação à "fonte de dados B", replicando o componente;
3. Não existe na "fonte de dados A" e é adicionado com recurso a novos atributos, então a operação de adição deste novo atributo é aplicada ao processo de transformação da "fonte de dados B", validando se os atributos usados fazem parte desta fonte.

Por sua vez, e imaginando que o atributo "CardId" era removido do esquema esperado, com esta metodologia esse atributo era identificado como sendo um atributo que deveria ser removido, e nestas situações, a remoção de atributos é feita apenas no final do processo de transformação, uma vez que não é viável remover este atributo sem antes efetuar todas as operações que forem necessárias, pois podem necessitar deste atributo para essas operações.

3.1.2 Diagrama de classes

No desenho desta solução foi criada uma estrutura de classes que permita que esta metodologia seja transversal a qualquer ferramenta de KDD, tal como é possível visualizar na Figura 14. Essa estrutura é constituída por uma classe responsável por interagir com o utilizador, mostrando e pedindo informações acerca do processo a ser seguido, para além de uma interface com os métodos que devem ser implementados pelas classes que a irão implementar.

A interface desenvolvida representa a metodologia desenvolvida. O método "GetListFields" é responsável por obter as informações relativas aos principais componentes envolvidos nesta metodologia, sendo eles: a fonte de dados original, o esquema pré-mineração e a nova fonte. Já o método "CreateScriptForMergeInputs" é responsável por identificar quais as diferenças entre as fontes de dados. O "CreateScriptForNewAttributes" é o método responsável por obter os componentes que foram adicionados e que realizam operações sobre a fonte de dados original. O "ReadNewAttributes" é um método responsável por criar os componentes destinados a resolver as diferenças identificadas no método "CreateScriptForMergeInputs". O método "ExcludeAttributes" é responsável por remover da nova fonte de dados os atributos que não têm correspondência com o resultado esperado. Já o "CreateNewNodes" é o método responsável por enviar para a *stream* tudo o que foi criado e processado pela solução.

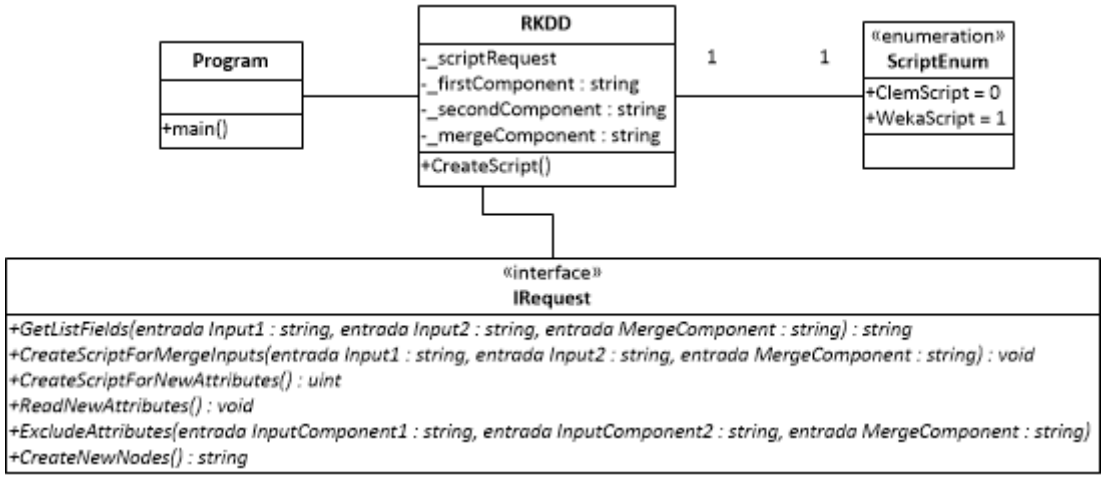

Figura 14 – Diagrama de classes – sem implementações da interface

O diagrama de classes é uma peça fundamental do desenho da solução, e como tal, no desenho desta solução foi criado um diagrama de classes relativo a cada uma das ferramentas de KDD escolhidas. No desenvolvimento, para cada uma das ferramentas, é

usado o polimorfismo, sendo a classe base a "IRequest" que define quais os métodos a serem implementados por cada uma das classes que implementem esta interface.

3.1.2.1 IBM SPSS Modeler

No diagrama de classes representado na Figura 15, encontram-se definidas as classes que são necessárias para a solução proposta, sendo que, no caso do *IBM SPSS Modeler*, a classe "ClemScriptRequest" implementa a interface definida e, por consequência, é responsável por implementar cada um dos passos da metodologia.

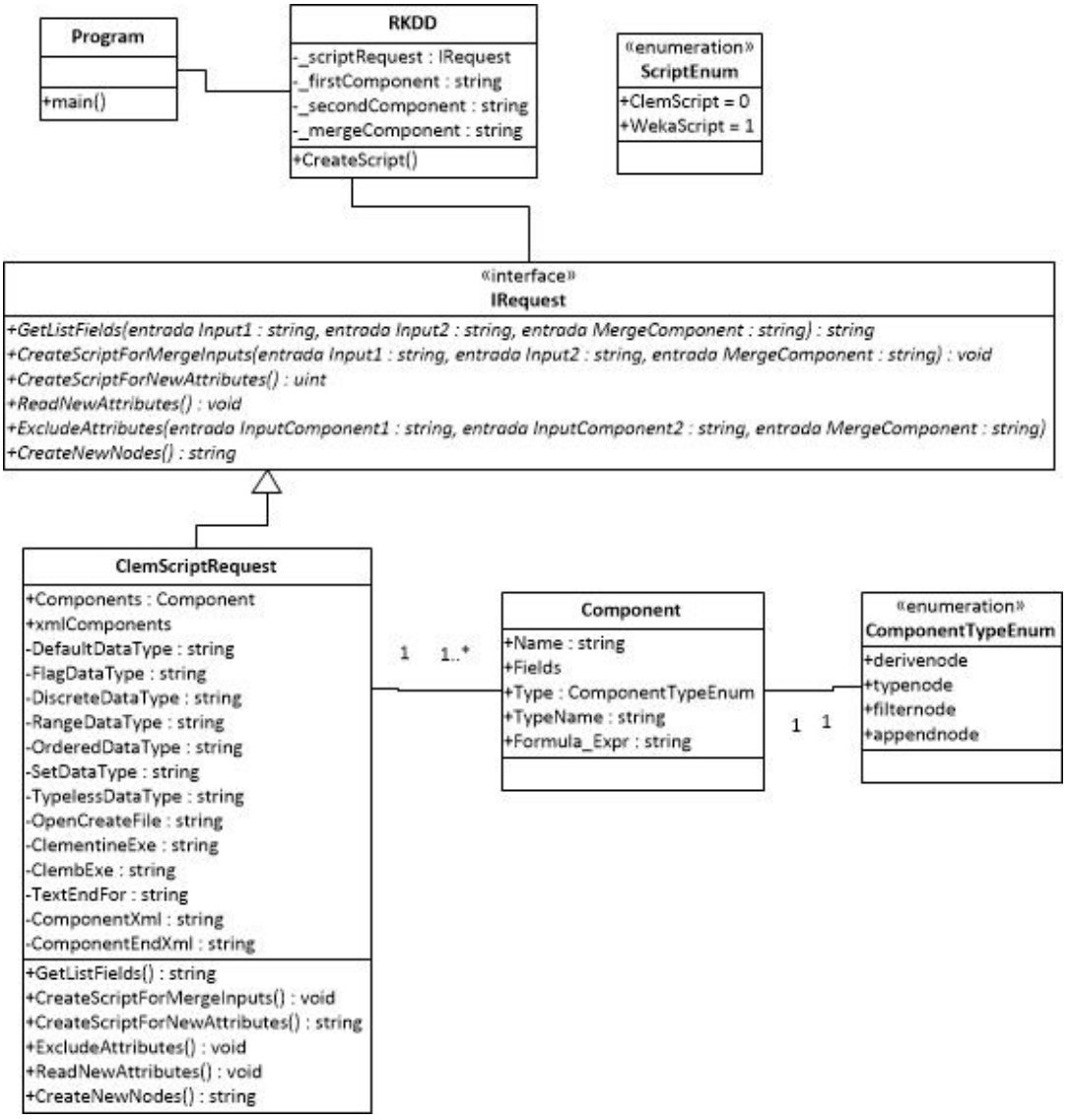

Figura 15 – Diagrama de classes relativa à implementação *IBM SPSS Modeler*

3.1.2.2 WEKA

No diagrama de classes ilustrado na Figura 16, é possível verificar as opções tomadas relativamente às técnicas e metodologias de engenharia de *software* adotadas no desenvolvimento relativo à ferramenta WEKA. Foi adotada a herança de classes entre a classe "WekaComponent" e as classes: "StringToNominal", "AddExpression", "CSVLoader" e "RemoveComponent". Para a leitura do ficheiro do WEKA foi adotado o padrão "*Singleton*" dada as iterações que eram necessárias com o ficheiro, sendo que com recurso a este padrão tornou-se mais simples aceder às propriedades do ficheiro, ou até mesmo na manipulação do mesmo, uma vez que reutiliza a mesma instância.

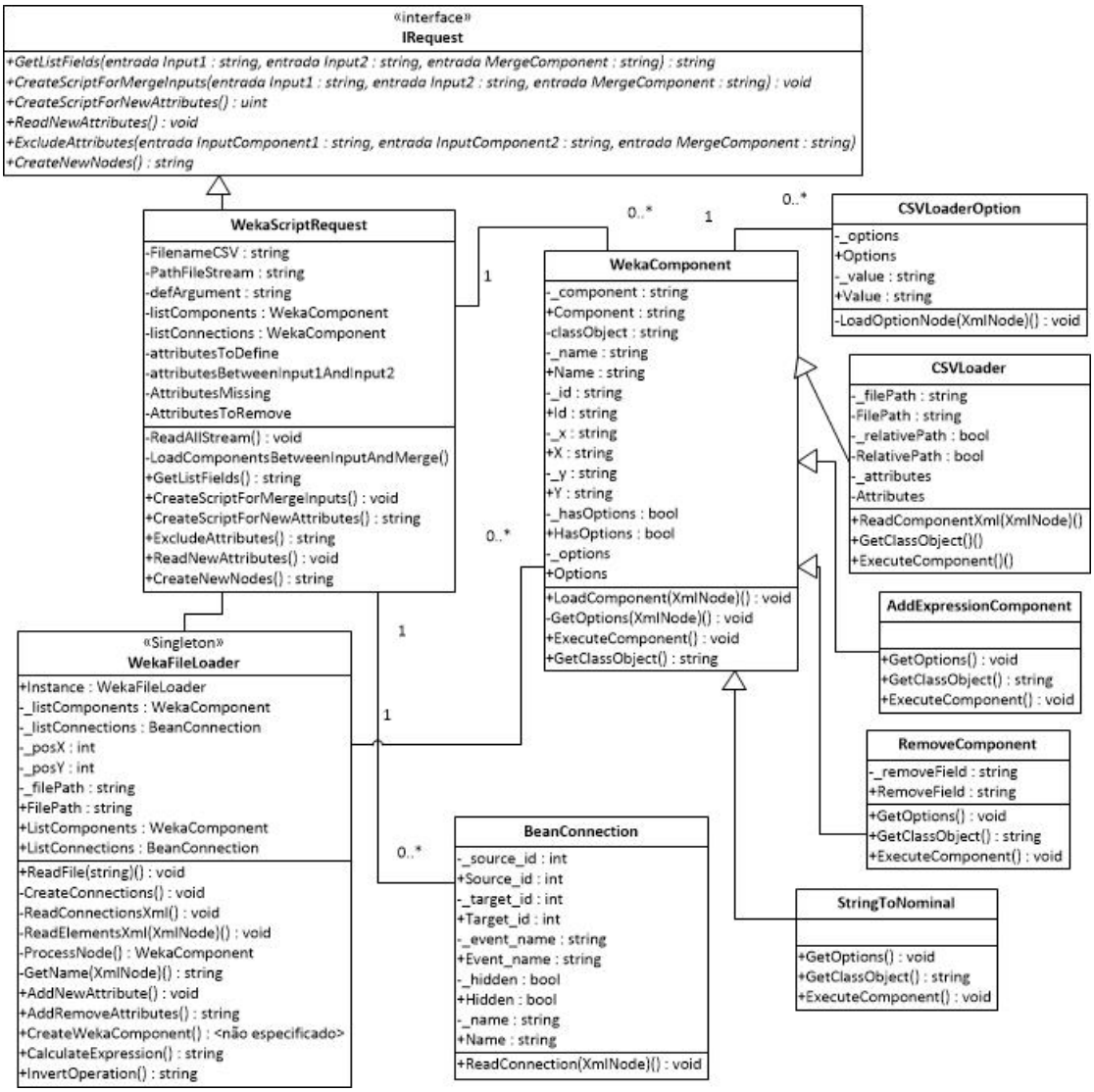

Figura 16 – Diagrama de classes da implementação em C#

3.1.3 Diagrama de sequência

No diagrama de sequência representado na Figura 17 está representado o *Core* da metodologia implementada. No diagrama encontra-se representado a sequência de pedidos que são efetuados independentemente do *software* que se opte para a realização da descoberta de conhecimento. Mas, neste diagrama em específico, está a ser usado o *IBM SPSS Modeler* como exemplo, recorrendo à API que o próprio disponibiliza e que permite que sejam executados blocos de código CLEM. Este código é gerado através da implementação da metodologia e que vai interagindo e processando as respostas a cada pedido efetuado.

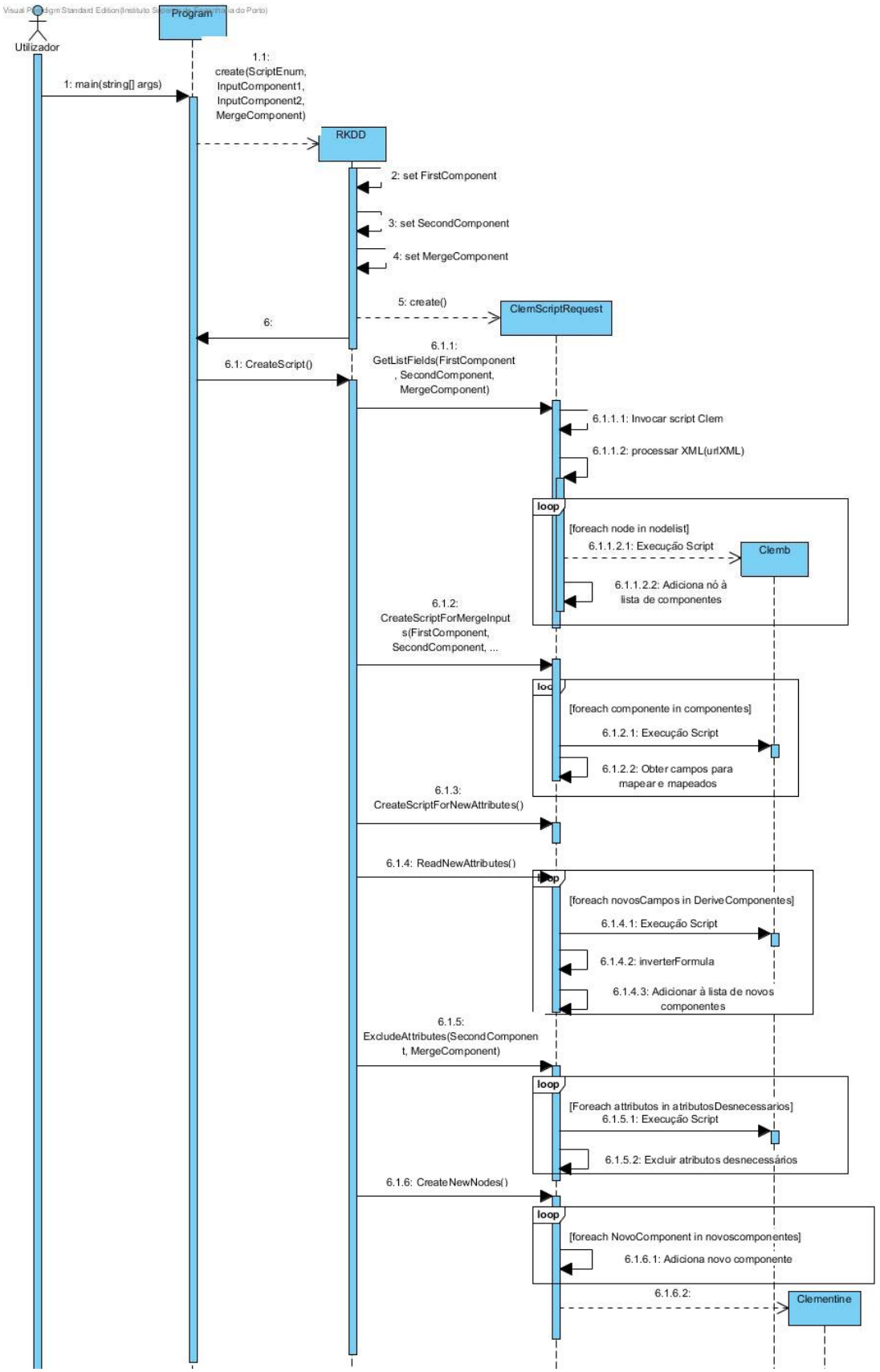

Figura 17 - Diagrama de Sequência – Comunicação com o *IBM SPSS Modeler*

3.2 Implementação da solução

Após ter sido feita a descrição da solução e delineada a estratégia a ser seguida, segue-se a implementação da mesma baseada em C# e recorrendo ao Visual Studio.

Nesta secção está presente o algoritmo usado de base para a implementação, a descrição dos diagramas de sequência, a estrutura da solução implementada, a identificação das limitações encontradas na implementação e os resultados que foram obtidos. Na implementação da solução foram usadas as ferramentas *IBM SPSS Modeler* e o Weka, por forma a demonstrar que é uma metodologia que não está dependente de uma ferramenta específica.

3.2.1 Algoritmo da Solução

Após o desenho da metodologia, e tendo em vista a criação de uma ferramenta com base nela, surge a necessidade de definir um algoritmo em pseudo-código (ver Código 4) para que esta seja mais facilmente entendida, e para que se consiga perceber os diversos passos pelos quais esta metodologia passa. Desta forma, este algoritmo permite ter a perceção das comparações feitas entre esquemas, assim como perceber qual a estratégia de implementação que deverá ser seguida.

```
Metodologia(EsquemaA, EsquemaB, EsquemaDestino)
Inicio
      Booleano encontrado = false;
      Lista atributosParaDefinir = new lista();
      Para (AtributoED em EsquemaDestino.Atributos) então
              Para (AtributoB em EsquemaB.Atributos) então
                     Se (AtributoB == AtributoED) então
                            Encontrado = true;
                            Pausa;
                     Fim-Se;
              Fim-Para
              Se (Encontrado == false) então
                     atributosParaDefinir.Adiciona(AtributoED);
              Fim-Se
              Encontrado = true;
       Fim-Para
       Encontrado = false;
       Para (AtributoA em EsquemaA.Atributos)   então
              Para (AtributoB em EsquemaB.Atributos)   então
                     Se (AtributoB == AtributoA) então
                            Pausa;
                     Fim-Se
              Fim-Para
              Se              (Encontrado              ==              false
      && !atributosParaDefinir.Existe(AtributoA)) então
                     atributosParaDefinir.Adiciona(AtributoA);
              Fim-Se
              Encontrado = true;
       Fim-Para
       Lista                    componentesTransformação                    =
      ObterComponentesTransformacao(EsquemaA,EsquemaDestino);
       Para (componente em componentesUsadosTransformação) então
              Para(AtributoPorDefinir em atributosParaDefinir) então
                     Se(componente.CriaAtributo == AtributoPorDefinir) então
                            Componente NovoComponente = new Componente();
                            NovoComponente.Operacao = componente.Operação;
                            AdicionaComponenteStream(NovoComponente);
                     Fim-Se
              Fim-Para
       Fim-Para
       Componente componenteRemoveAtributo = new Componente();
       Para (AtributoB em EsquemaB.Atributos)   então
              Para (AtributoED em EsquemaDestino.Atributos)   então
                     Se (AtributoB == AtributoED) então
                            Pausa;
                     Fim-Se
              Fim-Para
              Se (Encontrado == false) então
                     componenteRemoveAtributo.Remove(AtributoB);
              Fim-Se
              Encontrado = true;
       Fim-Para
       AdicionaComponenteStream(componenteRemoveAtributo);
Fim
```

Código 4 – Algoritmo desenhado para a metodologia

3.2.2 Diagramas de sequência

3.2.2.1 Implementação IBM SPSS Modeler

Como foi referido na secção 3.1.3 - Diagrama de sequência, a implementação relativa ao uso da ferramenta *IBM SPSS Modeler* já se encontra representada na Figura 17.

No código que é gerado para obter informações é adotado o XML como o tipo de ficheiro de resposta aos pedidos, uma vez que este permite estruturar a informação e ser reutilizado mais facilmente por outras aplicações com outros âmbitos de atuação.

Em relação às APIs disponibilizadas, estas são duas, sendo que o que as distingue é o comando de execução: o "*clementine*" e o "*clemb*". A invocação do primeiro irá executar os comandos, mas, com o ambiente gráfico do *IBM SPSS Modeler* aberto. No caso da segunda, esta executa todos os comandos sem recorrer a uma execução visual. Isto é, no primeiro caso quando se invoca um comando, este irá executar o *Clementine/IBM SPSS Modeler*, sendo que a partir desse momento todas as ações terão de ser efetuadas no ambiente gráfico. Por sua vez, o *Clemb* permite que sejam executados comandos sobre um determinado processo previamente criado, não guardando o estado atual de cada execução, o que se traduz numa ferramenta útil para recolha de informações, como por exemplo na recolha de informações relativas aos componentes existentes numa *stream*.

Neste caso o projeto teria de ser criado normalmente, sendo que esta solução implementada, materializando a metodologia proposta, apenas é usada aquando da adição de uma nova fonte, sendo essa também adicionada através da própria aplicação, ficando o tratamento dos dados dependente da execução.

3.2.2.2 Implementação WEKA

A implementação criada para o WEKA foi distinta, já que este não permite que exista uma comunicação externa com a aplicação, como é possível no *Clementine/IBM SPSS Modeler*. No entanto, e avaliando o ficheiro que é gerado ao guardar a *stream*, verificou-se que recorre a XML para armazenar a informação relativa a essa mesma *stream* que é criada no ambiente gráfico.

Uma vez que recorrem ao formato XML, isso tornou possível a leitura da *stream*. Como não é possível interagir com o WEKA, replicou-se os componentes na solução implementada, permitindo assim processar os componentes e obter as informações necessárias, tais como:

os atributos e as ligações entre os componentes. As ligações que constam no ficheiro XML permitem recriar o processo em C#.

Como é possível verificar na Figura 31 (que se encontra em anexo a este documento devido ao facto de ser extenso), existe uma leitura integral do ficheiro XML, para garantir que nas operações seguintes todas as informações se encontram disponíveis. Para tal, foi criada a classe "WekaFileLoader" que é responsável por manter a informação do ficheiro atualizada, e é através desta que são manipulados os componentes gráficos.

3.2.3 Estrutura da solução no Visual Studio

Com base no diagrama de classes que se encontra explicitado anteriormente, foi desenvolvida a solução em Visual Studio, como é possível ver na Figura 18, usando a linguagem de programação C#. Linguagem esta orientada a objetos, adequando-se ao desenvolvimento que foi necessário criar, uma vez que era pretendido criar e manipular objetos.

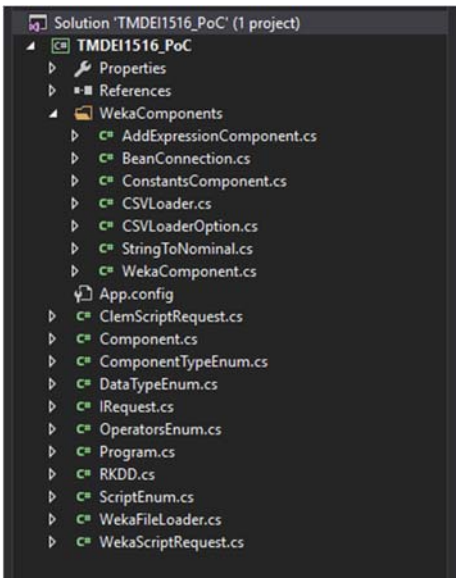

Figura 18 - Solução Implementada no Visual Studio

Para a implementação desta solução mostrou-se necessário converter o código-fonte do WEKA utilizando para tal uma ferramenta que se encontra disponível e de forma gratuita, o ikvmc (ver secção D - IKVM.NET), e que permite adicionar a solução do WEKA ao projeto como sendo uma biblioteca externa. Permitiu que fossem criados e executados os componentes, tal como estes se encontravam implementados na aplicação WEKA, permitindo por fim obter a informação necessária e crucial.

Com a inclusão do WEKA na solução como uma biblioteca, foi necessário criar um interpretador dos ficheiros (“WekaFileLoader”) gerados por esta aplicação que permite que os dados depois sejam usados para replicar os componentes internamente. Ao replicar os componentes com as configurações que estes têm, permite-se assim que se consiga interpretar o resultado final, neste caso identificar quais os atributos que são esperados no esquema pré-mineração.

A identificação das operações a serem realizadas sobre a nova fonte de dados são baseadas nos atributos do componente final e nos atributos da fonte original, comparando-os e extraindo a informação relevante para que seja possível aplicar as fases de pré-processamento e de transformação e dessa forma obter o esquema de dados pré-mineração.

3.2.4 Limitações e Problemas desta fase

Uma das limitações que é transversal às duas ferramentas utilizadas reside no facto de que nem sempre é possível inverter a operação de cálculo de um novo atributo adicionado à fonte de dados original, uma vez que está dependente da complexidade que essa operação pode ter. Como exemplo, pretende-se adicionar o atributo “Desconto” à nova fonte, e a equação terá de ser resolvida em ordem a “Desconto”:

Tabela 5 - Adaptação de uma operação em relação ao atributo esperado

Operação	**Novo atributo**	**Operação esperada**	**Resultado Alcançado**
TotalSemDesconto = Total/ ((100 - Desconto)/100)	Desconto	Desconto = (-100*Total + 100*TotalSemDesconto) / TotalSemDesconto	Insucesso
PrecoFinal = PrecoInicial - Desconto	Desconto	Desconto = PrecoInicial - PrecoFinal	Sucesso

No primeiro caso não se obteve sucesso no resultado uma vez que a operação utilizada continha uma complexidade superior ao que é suportado, neste momento, pela solução desenvolvida, já que apenas suporta operações simples, tal como ilustrado no segundo caso.

Com isto, uma das limitações é a complexidade que cada cálculo poderá exigir, podendo ser possível converter numa nova operação ou não. Deste modo, caso não seja

possível essa conversão, é mostrada uma mensagem durante a execução referente a isso. Esta é uma limitação a ser ultrapassada e que fará parte dos desenvolvimentos futuros.

Uma outra limitação, mas neste caso relativa ao *IBM SPSS Modeler*, é a necessidade de executar pelo menos uma vez o processo criado antes de executar a solução desenvolvida, já que os componentes que fazem parte da *stream* ainda não contêm a informação relativa ao esquema pré-mineração. Sem ser executado uma vez, os componentes apenas conhecem as suas funções e não o resultado das mesmas.

Ainda no *IBM SPSS Modeler*, existe uma outra limitação que está associada com o facto da aplicação não poder estar a ser executada, uma vez que quando é invocado o "clementine.exe" este só irá executar caso a ferramenta esteja encerrada. Caso contrário, não irá ser possível ver os novos componentes a surgirem.

No caso do WEKA existe uma limitação que é referente à indicação do componente que possui os atributos que constituem o esquema pré-mineração. Este não pode ser um componente de visualização uma vez que não possui uma forma de programaticamente se obter esses mesmos atributos. Terá de ser usado o último componente utilizado na fase de transformação, isto é, o utilizador terá de indicar esse componente aquando da execução da solução.

Por fim, outra limitação identificada e que é transversal às duas ferramentas reside no facto de não ser validado o tipo de atributo, sendo que a correspondência se restringe apenas ao nome que este possui, tendo o analista de ajustá-los antes da execução da solução desenvolvida.

3.2.5 Resultado da Implementação da Solução

Depois de apresentada toda a estrutura da implementação da solução, a seguir são mostrados os resultados da mesma. Utilizando o exemplo apresentado na secção 2.2 e recorrendo às ferramentas de KDD escolhidas, segue-se o resultado.

3.2.5.1 Implementação em IBM SPSS Modeler

Para as primeiras fases do *IBM SPSS Modeler* foram usados os seguintes tipos de componentes: ".Var File", "Derive", "Table" e "AppendOutput". Na Figura 19 é possível ver o processo definido para a primeira fonte, assim como a definição do novo atributo "BeforeBalance" baseado na expressão de cálculo "Amount+AfterBalance".

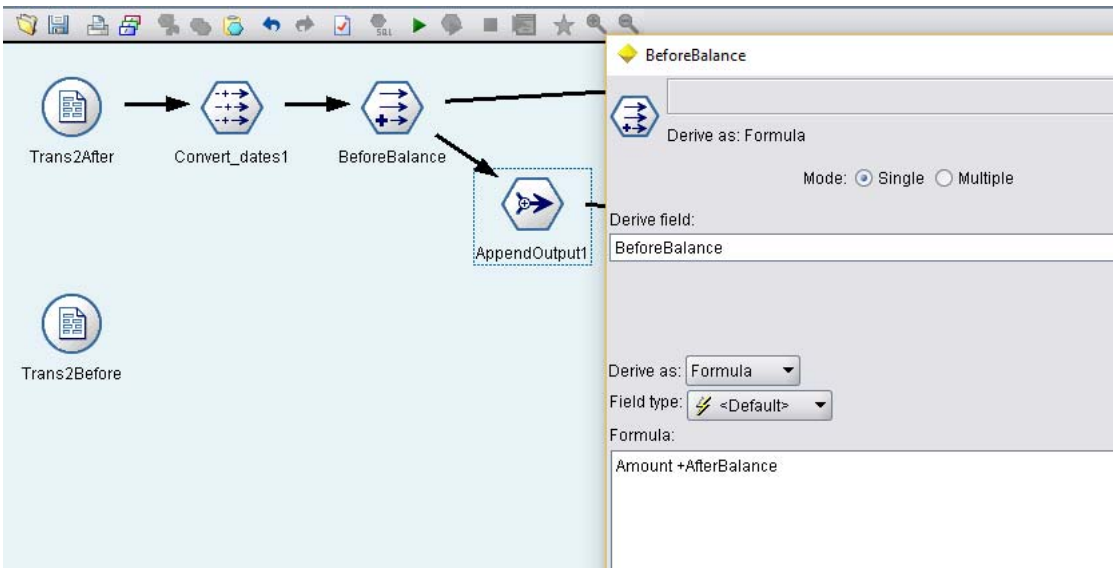

Figura 19 – *Stream* de exemplo utilizada com a definição de um atributo

Na Figura 20, por sua vez é a execução da solução, em que foi questionado ao utilizador qual era a ferramenta de KDD que este estava a utilizar, qual a localização do ficheiro, quais eram as fontes e qual era o componente da *stream* que possuía o esquema pré-mineração.

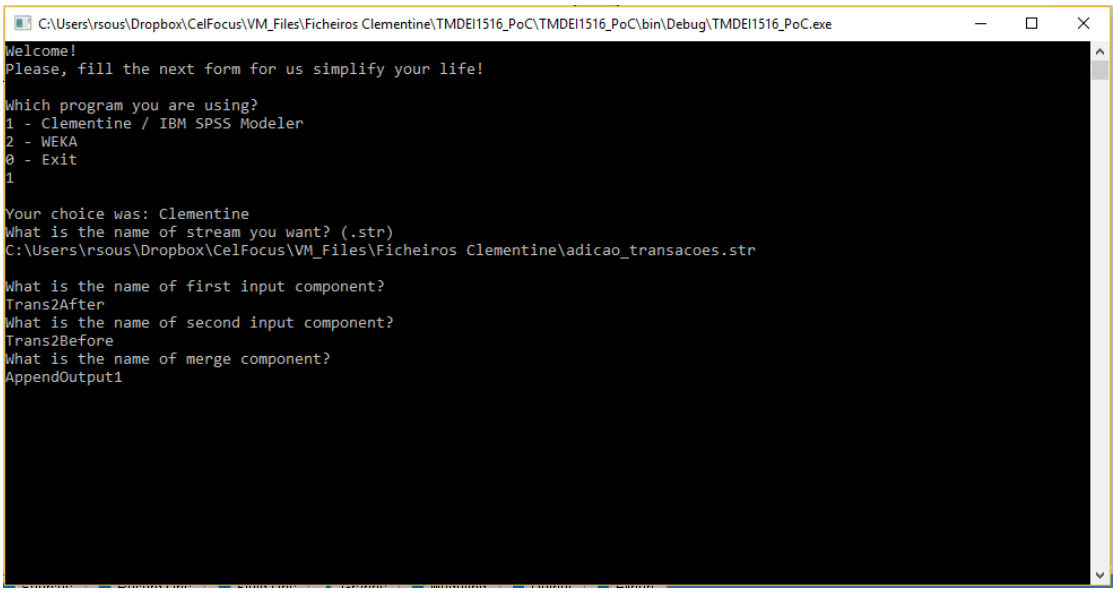

Figura 20 - Execução da solução implementada

De seguida, é apresentado o resultado final na Figura 21, refletindo as alterações que foram identificadas como necessárias e que foram depois colocadas na *stream*, assim como a expressão que é usada para calcular os valores do novo atributo, que se encontra no canto inferior direito da imagem.

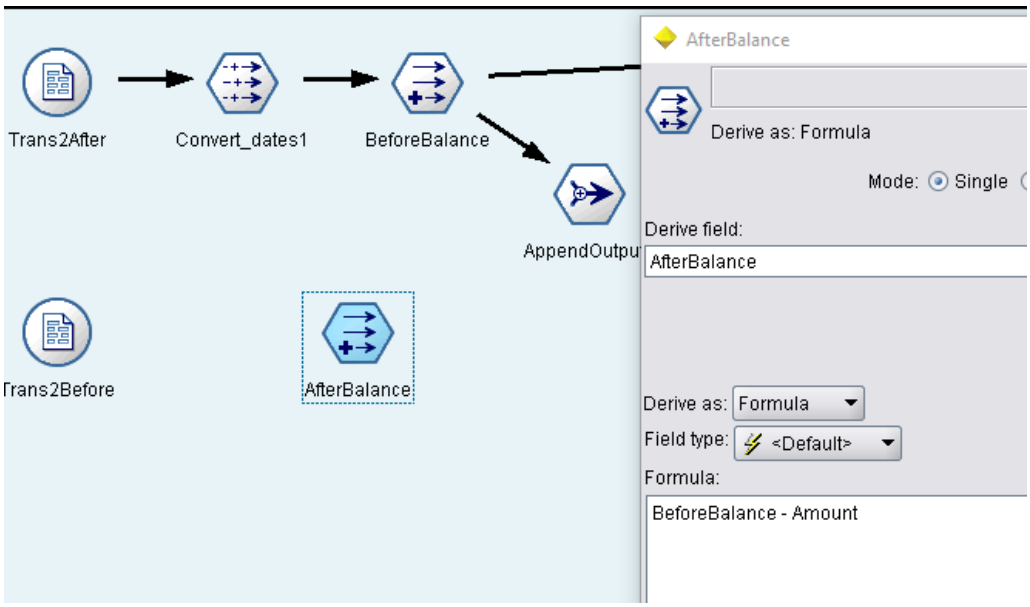

Figura 21 – Resultado da execução com o novo componente a ser adicionado.

3.2.5.2 Implementação em WEKA

No caso do WEKA a execução é muito semelhante, o que distingue é o tipo de componentes utilizados, que foram: "CSVLoader", "AddExpression", "TextViewer". Também neste caso é adicionado o mesmo atributo que foi adicionado no *IBM SPSS Modeler*, por sua vez a fórmula de cálculo é que é distinta, uma vez que no WEKA as fórmulas não se baseiam nos nomes dos atributos, mas nas posições que estes ocupam. Isto é, na Figura 22 está presente o componente "AddExpression" que representa a adição do atributo "BeforeBalance", sendo que a expressão utilizada foi a "a3+a4" que surge na Figura 23, em que o "a3" significa o atributo "Amount", e o "a4" o atributo "AfterBalance".

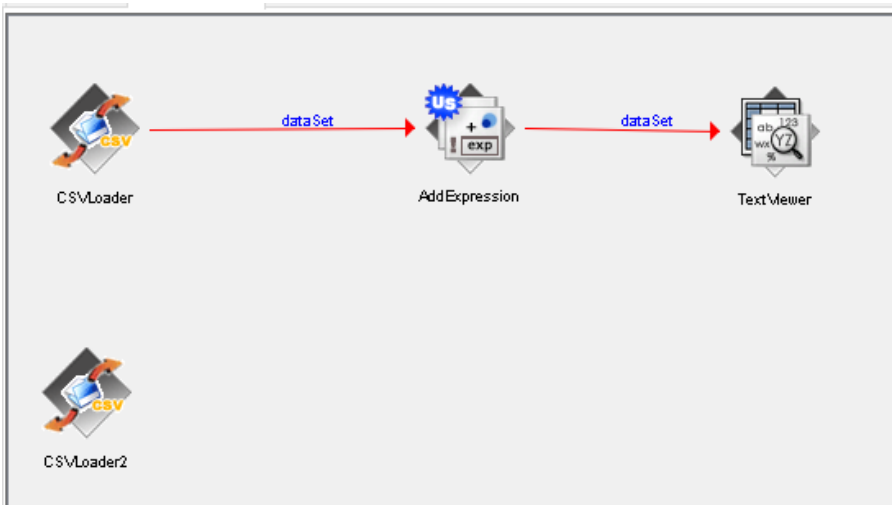

Figura 22 – *Stream* utilizada no WEKA

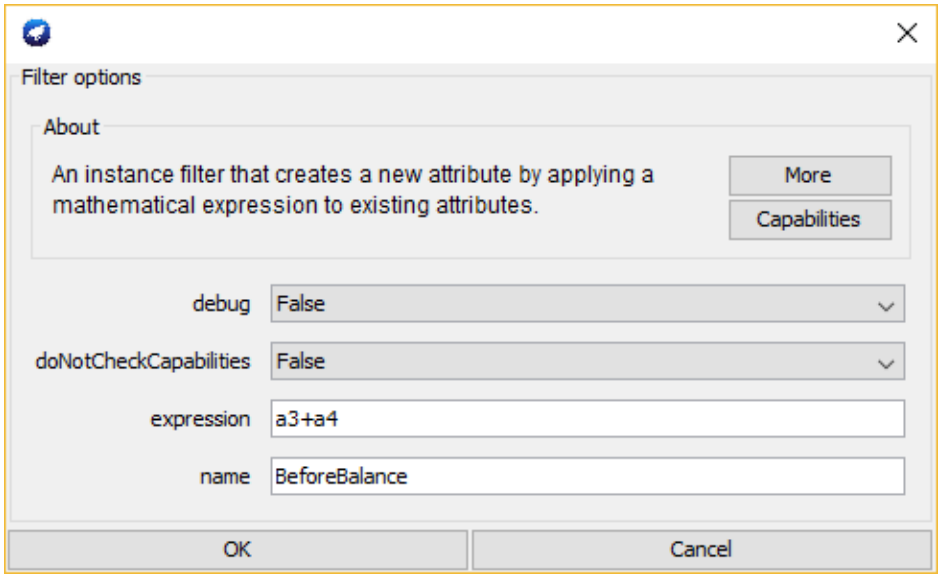

Figura 23 – Janela para definição do novo atributo

Na Figura 24 segue a execução da solução implementada para o WEKA, em que são efetuadas as mesmas questões que foram feitas para o *IBM SPSS Modeler*, questionando acerca da ferramenta, do ficheiro que contém a *stream,* e dos componentes que necessários.

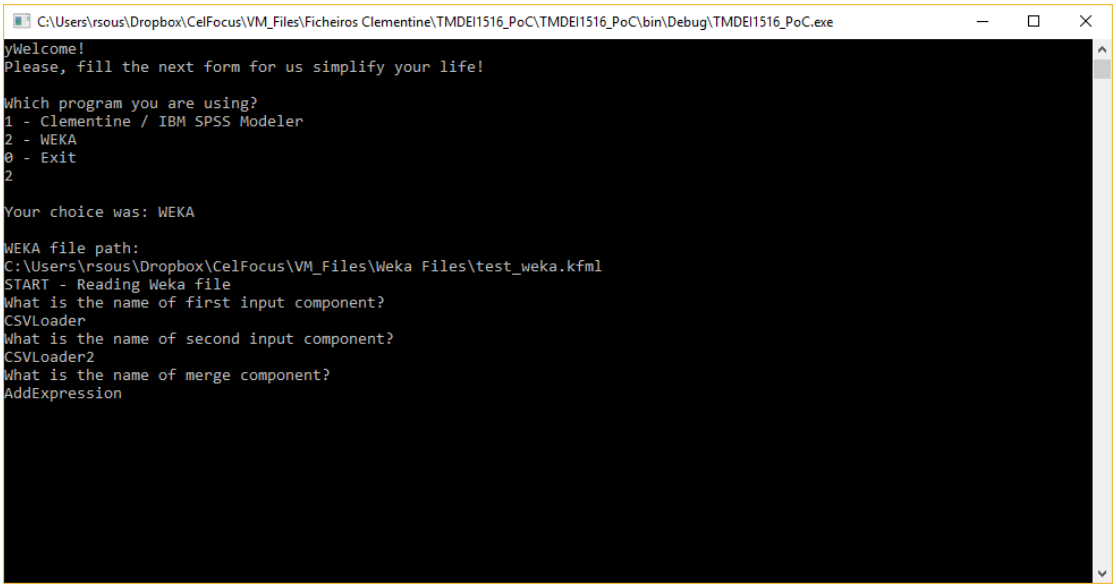

Figura 24 - Execução da componente na ferramenta WEKA

O resultado da execução encontra-se na Figura 25 em que é possível ver que um novo componente surge conectado à fonte de dados que se pretende manipular. Do lado direito está presente o nome do atributo que é necessário adicionar e a expressão utilizada no cálculo desse.

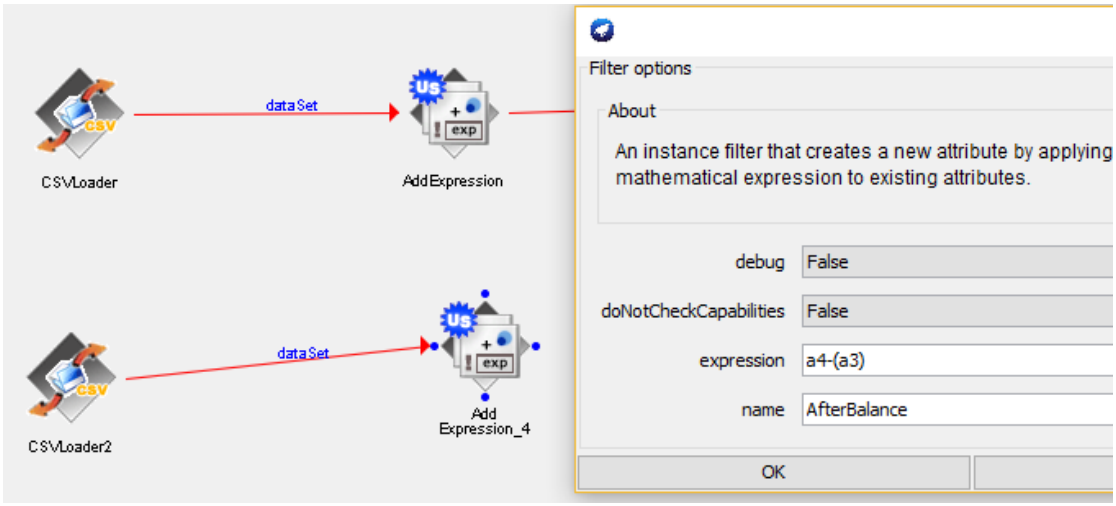

Figura 25 - *Stream* com o novo componente *AddExpression* acompanhado da sua definição

4 Testes

Neste capítulo é apresentado o processo de validação a que foi sujeita a implementação descrita anteriormente. São apresentados os casos de teste utilizados ao longo do trabalho e a análise de cobertura de funcionalidades para cada um deles. Da mesma forma que também são descritos os testes unitários implementados, assim como a análise de cobertura dos mesmos.

4.1 Testes Funcionais

Durante a implementação foram utilizados três casos de teste criados com o objetivo de cobrir a maior parte das funcionalidades de transformação de dados, incluindo a adição de novos componentes. Estes testes foram extremamente úteis para guiar a correta evolução da implementação das funcionalidades e validar a conclusão de cada uma destas.

Os casos de testes usados como validadores ao longo do tempo das novas funcionalidades encontram-se descritos a seguir. Estes suportaram o desenvolvimento para cada uma das ferramentas, tanto para o WEKA como para o *IBM SPSS Modeler*. Deste modo, a seguir também são apresentados os resultados obtidos para cada um destes.

Para o teste destas funcionalidades foi usado o exemplo dado na secção 2.2 e que é apresentado novamente a seguir, assumindo que o esquema A foi utilizado na implementação do processo, e que o esquema B é o novo esquema a ser transformado conforme as condições dos testes.

Tabela 6 – Esquemas de dados utilizados nos testes funcionais

Esquema de dados A	Esquema de dados B
CardID	CardID
Data	Data
Valor	Valor
SaldoFinal	SaldoAnterior

4.1.1 Adição de campo em falta na segunda fonte

O esquema de dados pré-mineração contém um campo que não está contemplado na segunda fonte de dados, pelo que esse campo terá de ser adicionado com base em operações efetuadas sobre a primeira fonte, uma vez que é expectável que seja capaz de manipular os cálculos usados e aplicá-los à nova fonte. Deste modo, o objetivo é adicionar o "SaldoAnterior".

Tabela 7 – Caso de teste 1

Funcionalidade	Adição de um campo em falta à segunda fonte
Pré-condições	1. Duas fontes e um componente com o esquema pré-mineração; 2. A segunda fonte não possui o novo campo ("SaldoAnterior"); 3. Possui os outros campos identificados no esquema da fonte A.
Procedimento	1. Introduzir o nome da primeira fonte; 2. Introduzir o nome da segunda fonte; 3. Introduzir o nome do componente que define o esquema final pré-mineração.
Resultado Esperado	1. O novo campo deverá ser adicionado à segunda fonte através de um novo componente.

Os resultados dos testes com base no caso de teste enunciado anteriormente são apresentados na Tabela 8.

Tabela 8 – Resultados do caso de teste 1

Ferramenta	Resultado	Observações
IBM SPSS Modeler	Não passou	Código enviado para a ferramenta continha erro na expressão.
IBM SPSS Modeler	Passou	Expressão já se encontrava correta.
WEKA	Não passou	Componente não foi adicionado porque não estava a ser adicionado ao XML do ficheiro do WEKA.
WEKA	Passou	Componente criado.

No caso dos testes que não passaram, foi corrigido o código. Isto é no caso do *IBM SPSS Modeler*, existia um erro na obtenção da expressão de cálculo do atributo, tendo sido ajustado o método responsável por calcular essa expressão. Por sua vez, no caso do WEKA, estava a ser criado o objeto no código, mas não estava a ser adicionado ao nó do XML.

4.1.2 Adição de novo campo inexistente em ambas as fontes

Por vezes é necessário refinar os atributos, aumentando o nível de granularidade. Este é um cenário válido, pelo que a adição de um novo campo a ambas as fontes é uma situação recorrente, tornando expectável que esse campo seja adicionado à segunda fonte da mesma forma que é adicionado à primeira. Como exemplo, tem-se as duas fontes mencionadas no início desta secção, e pretende-se adicionar o novo campo "imposto", sabendo que em todos os registos é a mesma taxa de imposto.

Tabela 9 – Caso e teste 2

Funcionalidade	Adição de um novo campo a ambas as fontes
Pré-condições	1. Duas fontes e um componente com o esquema pré-mineração; 2. Nenhuma das fontes possui este novo campo ("imposto"); 3. Possuem os outros campos identificados (os campos do exemplo);

	4. Existe um componente que adiciona o novo campo a um fluxo de dados.
Procedimento	1. Introduzir o nome da primeira fonte; 2. Introduzir o nome da segunda fonte; 3. Introduzir o nome do componente que define o esquema final pré-mineração.
Resultado Esperado	1. O novo campo deverá ser adicionado ao fluxo de dados relativo à segunda fonte.

Os resultados dos testes com base no caso de teste enunciado anteriormente são apresentados na Tabela 10.

Tabela 10 – Resultados dos testes do caso de teste 2

Ferramenta	**Resultado**	**Observações**
IBM SPSS Modeler	Passou	Componente adicionada corretamente.
WEKA	Passou	Componente adicionada corretamente.

4.1.3 Remoção de campo da segunda fonte

O terceiro caso de teste consiste na remoção de campos que são considerados desnecessários para o esquema pré-mineração. Deste modo estes campos são descartados do segundo esquema, apesar de existir a possibilidade deste ser usado no cálculo de outros campos e desse modo é descartado apenas no último passo, uma vez verificada a existência de campos desnecessários no esquema pré-mineração. Como exemplo, a remoção do campo "CardId" que é usado em ambas as fontes do exemplo inicial.

Tabela 11 – Caso de teste 3

Funcionalidade	Remoção de um campo da segunda fonte
Pré-condições	1. Duas fontes e um componente com o esquema pré-mineração;

	2. O componente responsável por unir as fontes não possui determinado campo; 3. O campo faz parte da segunda fonte.
Procedimento	1. Introduzir o nome da primeira fonte; 2. Introduzir o nome da segunda fonte; 3. Introduzir o nome do componente que define o esquema final pré-mineração.
Resultado Esperado	1. Deverá ser adicionado um novo componente para remover o campo do esquema da segunda fonte.

Os casos de teste foram sendo aplicados na implementação da solução produzindo resultados que confirmavam ou não a implementação da solução. Deste modo, na Tabela 11 apresentam-se os resultados obtidos para cada um dos testes.

Tabela 12 – Resultados dos testes ao caso de teste 3

Ferramenta	**Resultado**	**Observações**
IBM SPSS Modeler	Não passou	Adicionava o componente, mas não identificava qual o atributo a excluir.
IBM SPSS Modeler	Passou	Adicionou o componente corretamente.
WEKA	Passou	Adicionou ao XML o componente com o respetivo atributo para remover.

Neste teste, no caso do *IBM SPSS Modeler*, existia uma inconsistência com o nome dos campos que deviam ser removidos do esquema, sendo que estava a ser passado um valor vazio, sem caracteres.

Com isto, as principais funcionalidades requeridas foram implementadas com sucesso segundo os resultados dos testes realizados.

4.2 Testes unitários

Os testes unitários devem ser usados para validar as funcionalidades implementadas e se estas estão a ser implementadas corretamente, sendo que para cada conjunto de parâmetros de entrada, existe uma resposta que deverá ser dada pela funcionalidade e que deverá coincidir com o que é esperado.

Deste modo, os testes unitários implementados comtemplam para cada funcionalidade um conjunto de testes. Estes deverão assegurar que existe um tratamento específico para cada conjunto de parâmetros de entrada, garantindo que todas as situações são cobertas.

Os testes incidiram principalmente sobre as principais funcionalidades deste desenvolvimento, sendo que cobriram os casos de testes mencionados na secção anterior, avaliando diferentes parâmetros.

Tal como é possível observar na Figura 26 foram criados mais de 34 testes para garantir e validar a implementação das funcionalidades.

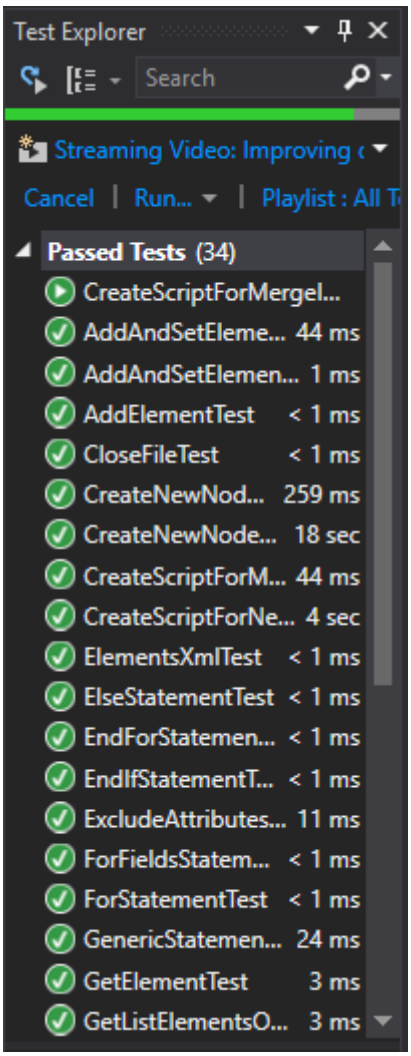

Figura 26 – Execução de todos os testes unitários implementados.

A seguir segue-se um dos testes unitários executados com a finalidade de garantir que a funcionalidade é executada com sucesso. Este teste é a uma das principais funcionalidades da metodologia, isto é, o teste à funcionalidade de encontrar quais os campos que se encontram em falta na nova fonte, em relação ao esquema de dados pré-mineração. Para a realização deste teste, foi incluída na solução uma *stream* de exemplo.

```
        [TestMethod()]
        public void CreateScriptForMergeInputsTest()
        {
            WekaScriptRequest req = new
WekaScriptRequest("test_weka.kfml");
            int res = req.ListComponents.Count;
            req.GetListFields(Input1, Input2, MergeComponent);
            req.CreateScriptForMergeInputs(Input1, Input2,
MergeComponent);
            Assert.AreEqual(req.attriToDefine.Count, 1);
            Assert.AreEqual(req.attributesbetweenInput1And2.Count, 1);
        }
```

Código 5 – Teste unitário à funcionalidade de extração dos campos em falta

Em relação ao resultado esperado, neste caso o que se pretendia é que fosse identificado um atributo que se encontra em falta na nova fonte e que este fosse adicionado à lista “attriToDefine”.

5 Experiências e avaliação da solução

Para validar a solução a ser desenvolvida, serão realizadas experiências relativamente ao tempo de execução da solução e do processo que atualmente tem de ser feito, para comparar efetivamente a eficácia e a vantagem desta mesma solução.

Como já foi referido anteriormente, o tempo de execução da operação será uma das maiores vantagens da solução, já que esta tem como objetivo otimizar um conjunto de tarefas que neste momento têm de ser executadas de forma independente uma das outras, e que têm de ser configuradas e executadas manualmente sempre que pretende aplicar o processo de descoberta de conhecimento num conjunto de dados similar, o que faz com que seja despendido demasiado tempo nessas operações.

5.1 Experiências realizadas

Para aferir a eficiência da solução implementada foi utilizado um conjunto de fontes, em que as mesmas foram submetidas ao processo manual e à solução desenvolvida. Foi utilizado um par de fontes em cada experiência realizada, sendo que uma é utilizada para a definição do processo e do esquema pré-mineração e a outra é submetida à solução implementada. Essas fontes encontram-se descritas na Tabela 13:

Tabela 13 – Lista de experiências a serem realizadas

Nº Experiência	*Esquema A*	*Esquema B*	*Esquema Pré-mineração*	*Operações a realizar*
1	*CartaoID; Data; Valor; SaldoAtual*	*CartaoID; Data; Valor; SaldoAnterior;*	*CartaoID; Data; Valor; SaldoAnterior; SaldoAtual;*	*Adicionar: -SaldoAtual*
2	*CartaoID; Data; Valor; Nome; NumeroItems; Desconto; NovoSaldoCartao*	*CartaoID; Data; Valor; Nome; NumeroItems; Desconto; SaldoCartaoAntigo*	*Data; Valor; NumeroItems; Desconto; SaldoCartaoAntigo; NovoSaldoCartao;*	*Adicionar: -Novo SaldoCartao Remover: -CartaoID -Nome*
3	*ClienteId; Data; CartaoId; Total; Desconto; IVA; NumeroItems.*	*ClienteId; Data; CartaoId; Total; TotalSemDesconto; IVA; NumeroItems; Hora; ClasseCliente.*	*ClienteId; Data; CartaoId; Total; Desconto; IVA; NumeroItems; PrecoMedio; TotalSemDesconto*	*Adicionar: -PrecoMedio -Desconto Remover: -Hora -ClasseCliente*

Na Tabela 14, e na Figura 27 estão presentes os resultados da execução das experiências mencionadas anteriormente que têm como objetivo comprovar a implementação da solução, assim como dar a conhecer se a solução é efetivamente uma resposta à necessidade levantada.

Tabela 14 – Experiências realizadas para comprovar viabilidade da solução

Ferramenta KDD	**Experiência**	**Duração processo Manual**	**Resultado da execução da solução**
IBM SPSS Modeler	*1*	*4 Minutos*	*≈ 28 Segundos*
	2	*5 Minutos*	*≈ 30 Segundos*
	3	*6 Minutos*	*≈ 31 Segundos*
WEKA	*1*	*4 Minutos*	*≈ 29 Segundos*
	2	*5 Minutos*	*≈31 Segundos*
	3	*6 Minutos*	*≈ 31 Segundos*

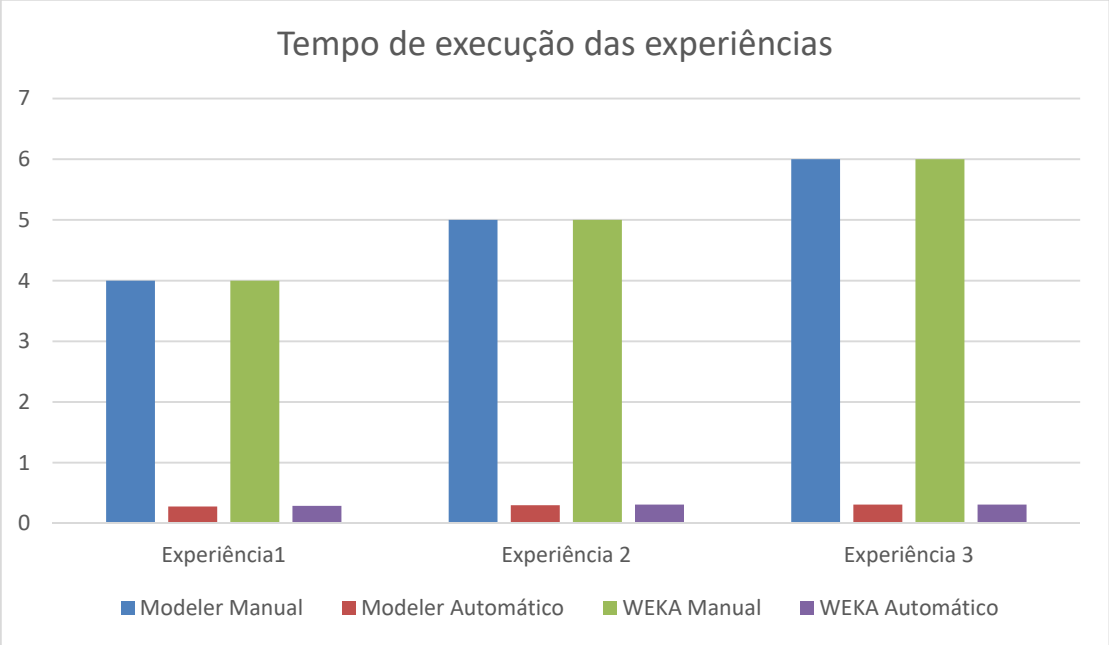

Figura 27 - Tempos de execução das experiências realizadas

Nota: Os resultados obtidos nestas experiências contemplam o preenchimento dos dados que são pedidos ao utilizador.

Segundo o que foi possível apurar, existe uma diferença substancial no tempo que é despendido na criação dos componentes de forma manual e o tempo que é despendido na criação dos mesmos, mas com recurso à solução desenvolvida. Esta permite logo à partida identificar quais os atributos que devem ser adicionados, e quais é que devem ser removidos do novo esquema de dados.

Neste momento é possível afirmar que quanto maior for o número de atributos de um esquema de dados, maior será a vantagem de recorrer a esta solução. O acréscimo do número de operações a serem tratadas de forma manual faz com que o analista despenda mais tempo, por sua vez, com recurso a esta solução, o aumento do número de operações nem sempre obriga a um maior tempo de execução (apesar de o tempo indicado comtemplar o preenchimento dos dados que são solicitados ao utilizador), como se pode ver pelo gráfico anterior em que existe um acréscimo mais acentuado na adição manual, do que com recurso à solução implementada.

6 Conclusão

Neste capítulo é feito um resumo dos principais resultados alcançados nesta dissertação e são apontadas algumas limitações e trabalho a ser desenvolvido no futuro para colmatar essas limitações.

6.1 Resultados

Neste trabalho sobre a reestruturação/adaptação de um processo de descoberta de conhecimento, foi realizada uma análise sobre como seria possível automatizar a reutilização de um processo de descoberta de conhecimento previamente definido, tendo em conta a necessidade inicial de reduzir o tempo despendido.

Os objetivos propostos inicialmente foram atingidos com sucesso, tendo-se solucionado a necessidade inicial, bem como de outros problemas detetados com o decorrer do trabalho.

A criação de uma nova metodologia era um dos objetivos propostos e que foi alcançado. Esta permite reutilizar um processo de descoberta de conhecimento em novas fontes tirando partindo da sua configuração inicial, reaproveitando ações que o analista efetuou na criação desse processo.

Com a criação dessa metodologia, foi possível alcançar o segundo objetivo estabelecido, uma vez que com recurso a essa o analista dispensa menos tempo na reconfiguração/adaptação. Para tal, foi criada uma ferramenta informática que implementa os passos definidos pela metodologia criada, readaptando assim o KDD utilizado.

A implementação desta metodologia foi exposta a testes que permitiram validar a metodologia criada de acordo com os testes efetuados, comprovando assim que esta metodologia permite reutilizar um processo de descoberta de conhecimento. Uma vez que os testes efetuados revelam uma melhoria no tempo despendido pelos analistas, comparativamente com o método manual tradicionalmente realizado por estes, e como foi automatizado este processo, logo não é suscetível à ocorrência de erros.

Ao longo desta dissertação houve um confronto entre vários desafios que correspondiam a limitações das ferramentas usadas como base para os desenvolvimentos efetuados. Estas limitações restringiam-se a problemas de interação entre a componente desenvolvida e as *streams* usadas por essas ferramentas. Estas limitações foram ultrapassadas num dos casos através da leitura do ficheiro da *stream*, e no outro, através da utilização de scripts em ClemScript.

Assim pode-se afirmar que esta solução implementada pode ter um impacto importante para os analistas, uma vez que permite reduzir o tempo que estes necessitam de despender e permite que possam reutilizar trabalho já desenvolvido anteriormente.

6.2 Limitações

Neste trabalho foram encontradas algumas limitações ao nível do desenvolvimento que provocaram a mudança da estratégia de desenvolvimento inicial. Mudança essa que impôs alterações ao nível da interação com as ferramentas.

As limitações foram ultrapassadas, sendo que no final, apenas restaram limitações em relação à solução implementada, uma vez que esta não permite que sejam escolhidos determinados tipos de nós de uma *stream* do WEKA, por estes não suportarem determinadas funções. Como tal, não podem ser usados para que sejam obtidas determinadas informações, como é o caso dos nós de visualização que não possuem informação acerca dos atributos relativos ao esquema pré-mineração, isto é, este tipo de nós não possuem métodos que devolvam os atributos que constituem o esquema. No entanto, esta limitação é perfeitamente ultrapassável, já que é possível prever essas informações através do último nó da fase de transformação que é usado e que deverá ser indicado no início da execução da solução.

No caso do *IBM SPSS Modeler*, a limitação é ao nível do processamento de informação, uma vez que exige que a *stream* não esteja a ser utilizada para que seja possível executar as ações pretendidas, e deste modo, que a metodologia possa ser aplicada.

Uma limitação que é transversal a qualquer uma das ferramentas utilizadas é o facto da solução implementada não ser capaz de efetuar determinadas transformações, devido à sua complexidade. Assim como a transformação de um atributo que se encontra num formato e que, no entanto, era expectável que o formato fosse outro.

Como a implementação efetuada consiste numa prova do conceito, muitas das funcionalidades que se encontram disponíveis nas ferramentas não se encontram implementadas, no entanto, esta prova de conceito confirma que a metodologia desenhada e implementada é válida e útil.

Uma limitação desta solução é o mapeamento de atributos, uma vez que não foram implementados mecanismos que permitam detetar que determinado atributo é o mesmo mas com designação diferente, sendo que apenas identifica que dois atributos são iguais se ambos tiverem a mesma designação.

6.3 Trabalho Futuro

Apesar de terem sido alcançados os objetivos propostos e dos resultados obtidos serem positivos, continuam ainda alguns problemas em aberto, tal como foi mencionado anteriormente, assim como algumas melhorias que podem serem efetuadas através de novas funcionalidades.

Como trabalho futuro, o expectável é que seja melhorada a introdução das informações iniciais pedidas ao utilizador, isto é, a introdução dos nomes dos componentes seja através da seleção a partir de uma lista e não da introdução manual do nome dos componentes.

Principalmente no caso do WEKA, o trabalho futuro irá incidir na criação de novos componentes que permitam a manipulação de diversos tipos de fontes de dados, assim como a manipulação dos atributos dessas fontes.

No caso da adição de novos atributos com base nas operações realizadas anteriormente, o objetivo será num trabalho futuro aumentar a capacidade de processar operações mais complexas.

Por fim, um dos próximos passos a ser desenvolvido é a criação de uma interface gráfica que permita fazer a gestão de tudo isto de uma forma visual, isto é, indicando numa

interface quais os componentes e o ficheiro, e visualizar os eventos pelos quais esta metodologia vai passando.

Referências Bibliográficas

Bernstein, P.A., et al. 2004. Industrial-Strength Schema Matching. *ACM SIGMOD Record 33.* 2004, pp. 38-43.

Bernstein, Philip A, Madhavan, Jayant e Rahm, Erhard. 2011. Generic Schema Matching, Ten Years Later. 2011.

Bernstein, Philip A., Madhavan, Jayant e Rahm, Erhard. 2001. Generic Schema Matching with Cupid. Proc. VLDB. 2001, pp. 49-58.

Conceptualising 'Value for the Customer': An Attributional, Structural and Dispositional Analysis. **Woodall, Tony. 2003.** 2003, Academy of Marketing Science Review.

Do, H.H. e Rahm, E. 2007. Matching large schemas: Approaches and evaluation. *Inf. Syst. 32(6).* 2007, pp. 857-885.

Elmeleegy, H., M. Ouzzani, and A.K. Elmagarmid. 2008. Usage-Based Schema Matching. Proc. ICDE. 2008, pp. 20-29.

Fayyad, Usama, Piatetsky-Shapiro, Gregory e Smyth, Padhraic. 1996. The KDD Process for Extracting Useful Knowledge from Volumes of Data. novembro de 1996.

Frawley, William J., Piatetsky-Shapiro, Gregory e Matheus, Christopher J. 1991. *Knowledge Discovery in Databases.* 1991.

Frijters, Jeroen. 2010. *IKVM.NET.* [Online] 2010. http://www.ikvm.net/index.html.

Gal, A. 2011. Enhancing the capabilities of attribute correspondences. In: Z. Bellahsene, A. Bonifati, E. Rahm (eds), Schema Matching and Mapping. s.l. : Springer, 2011, pp. 53-74.

Gross, A., et al. 2010. On Matching Large Life Science Ontologies in Parallel. Proc. 7th Int. Conf. Data Integration in the Life Sciences (DILS). s.l. : Springer LNCS 6254, 2010, pp. 35-49.

Grossman, Robert, et al. 2002. The Management and Mining of Multiple Predictive Models Using the Predictive Modeling Markup Language (PMML). *Information and Software Technology.* abril de 2002, pp. 589-595.

Gruber, Thomas R. 1993. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition.* 1993, pp. 199-220.

Hand, D. J., Mannila, Heikki e Smyth, Padhraic. 2001. *Principles of Data Mining.* s.l. : MIT Press, 2001.

He, B. e Chang, K. C.-C. 2003. Statistical Schema Matching across Web Query Interfaces. *Proc. SIGMOD.* 2003, pp. 217-228.

Heinze, Justin. 2014. History of Business Intelligence. [Online] 26 de setembro de 2014. https://www.betterbuys.com/bi/history-of-business-intelligence/.

IBM ;. 2016. SPSS Modeler. [Online] IBM, 2016. [Citação: 14 de janeiro de 2016.] http://www-01.ibm.com/software/analytics/spss/products/modeler/features.html.

IBM Knowledge Center. 2012. About CLEM. [Online] 2012. https://www.ibm.com/support/knowledgecenter/SS3RA7_15.0.0/com.ibm.spss.modeler.help/introduction_to_clem.htm.

IBM;. 2016. SPSS Modeler. [Online] IBM, 2016. [Citação: 15 de dezembro de 2015.] http://www-01.ibm.com/software/analytics/spss/products/modeler/index.html.

KNIME. 2016. KNIME Analytics Platform. [Online] 2016. https://www.knime.org/knime.

Knime. 2016. KNIME Open Source Story. [Online] 2016. https://www.knime.org/knime-open-source-story.

Li, J., J.Tang, Y. Li, and Q. Luo. RiMOM. 2009. A Dynamic Multistrategy Ontology Alignment Framework. IEEE Trans. Knowl. Data Eng. 21(8). 2009, pp. 1218-1232.

Lodder, Arno R. e Oskamp, Anja. 2006. *Information Technology and Lawyers: Advanced Technology in the Legal Domain, from Challenges to Daily Routine.* s.l. : Springer, 2006.

Madhavan, J., et al. 2005. Corpus-based Schema Matching. *Proc. ICDE.* 2005, pp. 57-68.

MaRS. 2002. *Crafting Your Value Proposition.* s.l. : MaRS Discovery District, 2002.

Marsh, Jennifer. 2014. Knowledge Discovery in Databases: 9 Steps to Success. [Online] 1 de abril de 2014. https://blog.udemy.com/knowledge-discovery-in-databases/.

Norton, M. Jay. 1999. *Knowledge Discovery in Databases.* 1999.

Perez-Rey, David, Anguita, Alberto e Crespo, Jose. 2006. OntoDataClean: Ontology-Based Integration and Preprocessing of Distributed Data. 2006.

Phillips, Joseph e Buchanan, Bruce G. 2001. Ontology-Guided Knowledge Discovery in Databases. 2001, pp. 123-130.

Quick ontology matching. **M., Ehrig e Staab, S. 2004.** 2004. Int. Conf. Semantic Web (ICSW).

Raisinghani, Mahesh. 2004. *Reference & Research Book News. August 2004, Vol. 19 Issue 3, p103, 1 p.* USA : Book News, Inc., 2004. p. 133.

Scatty. 1991. 1991.

Semantic Annotation and Services For KDD Tools Sharing and Reuse. **Diamantini, Claudia e Potena, Domenico. 2008.** 2008. 2008 IEEE International Conference on Data Mining Workshops.

WEKA. 2016. FracPete's Projects. *WEKA.* [Online] Maio de 2016. http://open.fracpete.org/projects/weka/.

Weka. 2016. Weka 3 - Data Mining with Open Source Machine Learning Software in Java. [Online] 2016. http://www.cs.waikato.ac.nz/ml/weka/index.html.

WEKA;. 2016. Machine Learning Group. [Online] 2016. http://www.cs.waikato.ac.nz/ml/index.html.

Wirth, Rüdiger e Hipp, Jochen. 2016. *CRISP-DM: Towards a Standard Process Model for Data.* Ulm, Germany : s.n., 2016.

Zhang, Shichao, Wu, Xindong e Zhang, Chengqi. 2003. Multi-Database Mining. junho de 2003.

Anexos

A Mineração de dados

Com uma maior recolha de dados as base de dados também acabaram por crescer, isto deveu-se à recolha de dados relativos às transações efetuadas com os cartões de crédito, ou informações relativas às chamadas telefónicas, por exemplo. Com isto o interesse começou a crescer e ao mesmo tempo a necessidade de extrair informações a partir desses mesmos dados, tentando encontrar possíveis padrões que interligassem determinados dados.

Mineração de dados é a análise de um conjunto de dados observacionais para encontrar relações e resumir os dados em novas formas em que são ambos percetíveis e uteis para o proprietário dos mesmos (Hand, et al., 2001).

As relações e os resumos derivados através da aplicação da mineração dos dados são frequentemente referenciados como modelos ou padrões. Sendo que o processo de mineração dos dados não tem qualquer influência na recolha dos mesmos, não estando assim formatados para responder a um conjunto limitado de questões. A mineração de dados é referenciado como sendo análise de dados "secundário".

Segundo os autores do livro "Principles of Data Mining" (Hand, et al., 2001), o conjunto de dados devem ser extensos, e quando estes são extensos novos problemas surgem. Estes problemas podem estar relacionados desde os dados não seguirem a mesma formatação ou então qual o nível de profundidade que se deve adotar, ou ate mesmo que período deve ser analisado, e quais os dados que estabelecem padrões que representam a realidade, ou não. Para ser realizado o processo de mineração de dados é extraído um subconjunto dos dados, definidos como uma amostra do conjunto.

Mineração de dados surge no contexto da descoberta do conhecimento nas bases de dados (KDD). A descoberta do conhecimento surgiu no campo de pesquisa da Inteligência Artificial (AI). Este processo subdivide-se em várias fases: seleção dos dados-alvo, processamento dos mesmos, efetuar transformações sobre eles, caso exista essa necessidade, aplicar técnicas de mineração de dados por forma a extrair padrões e relações entre os dados, interpretando e avaliando a informação estruturada recolhida.

O processo de procura de relações entre os dados dentro de um conjunto de dados envolve vários passos:

- Determinar a natureza e a estrutura dos dados a ser usada;

- Decidir como quantificar e comparar bem como representações diferentes ajustam os dados
- Escolher um processo algorítmico para otimizar a função de pontuação;
- Decidir quais os requisitos que são necessários para implementar o algoritmo eficientemente.

Mineração de dados é definido como sendo um exercício interdisciplinar. O que se torna difícil definir fronteiras claras entre o que é inteligência artificial, estatística, aprendizagem maquina, reconhecimento de padrões e visualização e mineração de dados.

B Conjunto de dados[13]

"D*ata set* é um conjunto de medidas tomadas a partir de alguns ambientes ou processos (Hand, et al., 2001)".

O conjunto de dados é constituído um conjunto de objetos do mesmo tipo, logo estes partilham entre si o mesmo conjunto de medidas, e transpondo esta informação para uma matriz, então cada linha representa um objeto e o valor para cada uma das p medidas. E por medidas entenda-se valores que permitam descrever o objeto e se possível agrupa-los conforme essa medida.

B.1.1 Modelos e Padrões

Durante o exercício de mineração de dados procura-se diferentes tipos de representações que podem ser caracterizadas de várias maneiras. Sendo que a caracterização é a distinção entre o modelo global e o modelo local.

Um modelo pode ter a formula Y = aX +c sendo Y e X as variáveis e a e c as parâmetros do modelo determinados durante a execução da mineração de dados. Esta fórmula é linear, desde que Y seja uma função linear de X. O conceito linear dado pelo autor é diferente do conceito linear associado à estatística, já que um modelo é linear se este é uma função linear dos parâmetros (Hand, et al., 2001).

[13] Conjunto de dados – traduzido do inglês Data Set

Relativamente aos padrões, um padrão local descreve a estrutura relacionada com uma parte pequena dos dados ou a um intervalo de tempo em que os dados tenham ocorrido. Mas por vezes nesse conjunto de dados recolhidos podem existir alguns registos diferentes da maioria, o que por vezes pode levar a que modelos globais e padrões locais sejam vistos como os lados opostos da mesma moeda, isto é, ambos podem indicar padrões e modelos opostos, o que nem sempre é favorável ou correto, já que os dados podem-se aplicar a um esquema de dados, mas, não se aplicar a mais nenhum. Com isto, também é possível extrair possíveis anomalias, como fraudes em bancos, por exemplo.

O objetivo final é consegui construir a fórmula representativa dos dados e atribuir valores às constantes da mesma, conforme os dados recolhidos, por forma a ser possível distinguir padrões e que tipo de comportamento irá resultar.

A distinção entre modelos e padrões é útil, mas nem sempre é fácil decidir se é um modelo ou um padrão.

B.1.2 Mineração de dados em Múltiplas Fontes de dados

Perante casos em que existem duas ou mais fontes de dados, e contendo estas dados distintos, existem algumas abordagens por forma a ser possível realizar mineração de dados sobre esses dados.

B.1.2.1 Análise de Padrão Local[14]

Dependendo do número de fontes de dados, os padrões nas múltiplas bases de dados poderiam ser classificados em 3 categorias, tais como: padrões locais, padrões globais e padrões que não são locais nem globais. Se um padrão é baseado apenas numa base de dados, é definido como padrão local. Este tipo de padrões são uteis para uma análise de dados e uma tomada de decisão local (Abhikari and Rao 2008b; Wu et al. 2005). Por sua vez, os padrões globais são baseados em todas as bases de dados tidas em consideração. São uteis para análises globais dos dados (Abhikari and Rao 2008a; Wu and Zhang 2003). Um modo conveniente para extrair padrões globais é extrair os padrões locais de cada base de dados, e então analisar todos os padrões locais para sintetizar os padrões globais (Hand, et al., 2001).

[14] Análise de Padrão Local – traduzido do inglês Local Pattern analysis

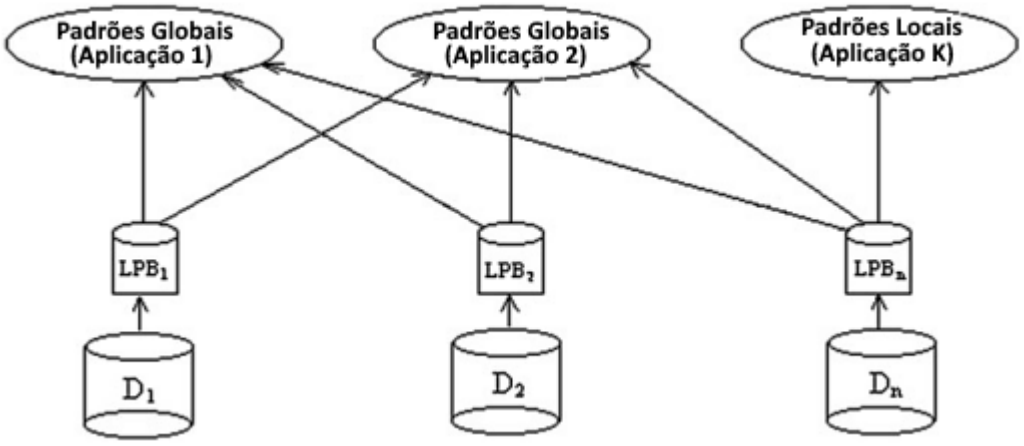

Figura 28 – Extração de padrões em múltipla base de dados com recurso a análise de padrões locais

Como foi referido descrito pelo auto do livro (Hand, et al., 2001), existe um primeiro passo que consiste em extrair os dados da base de dados local e identificar quais os padrões. E só depois de recolher todos os padrões relativos a cada base de dados, é que se irá juntar os padrões para se conseguir extrair os padrões globais. Com isto, é possível identificar quais os padrões locais e no caso das base de dados se referirem a comércio por exemplo, ser possível ajustar as promoções a cada loja ou região, ou então identificar padrões que se aplicam a todas as base de dados e tomar uma decisão global, e não específica (Hand, et al., 2001).

B.1.2.2 Amostragem[15]

Quando estamos perante um ambiente de múltipla base de dados, cada ramo representa uma base de dados, e a recolha dos dados a partir de todos os ramos pode tornar-se numa operação muito extensa e demorada. E com recurso à técnica de mineração de dados tradicional baseada em repositórios com enorme volume de dados, está provado que esta é mais difícil de se aplicar. Ao extrairmos dados aproximados de base de dados com elevado volume, estes devem ser adequados para a maioria das aplicações da tomada de decisão. Tais aplicações podem ser necessárias para ajudar a tomada de decisão de uma forma rápida. Para isto, recorre-se à amostragem, uma técnica que se define como a recolha de um conjunto de itens frequentes numa base de dados e que é considerado representativo de uma amostra da base de dados. Assim, permite que uma base de dados seja analisada através da análise dos itens frequentes numa representativa amostra de dados.

[15] Amostragem – traduzido do inglês Sampling

Com recurso à amostragem e à análise de padrões locais poderá ser uma técnica útil para extrair dados de várias bases de dados.

Resumidamente, a amostragem é uma técnica de mineração de dados que permite selecionar e extrair dados que são representativos de um conjunto de dados, identificando assim padrões e tendências nos dados recolhidos.

B.1.2.3 Re-mineração

Para a realização de mineração de dados em várias bases de dados pode-se aplicar um algoritmo de partição (Savasere et al. 1995). Este algoritmo está desenhado para extrair uma base de dados enorme através do particionamento.

A re-mineração consiste em examinar os dados 2 vezes. A base de dados é dividida em partições disjuntas, onde cada partição é pequena suficiente para caber na memória. Assim, da primeira vez que examina os dados, o algoritmo lê cada partição e processa a frequência local de todos os itens frequentes nessa partição usando o algoritmo *a priori*. Por sua vez, na segunda vez que examina os dados, o algoritmo contabiliza o número de vezes que determinado item frequente ocorre na base de dados completa. Replicando este processo para múltipla base de dados, uma partição representa uma base de dados local.

Contudo, este processo pode ser uma solução custosa para a realização de mineração em múltiplas bases de dados, uma vez que cada instância local requer ser examinada duas vezes. Durante a segunda examinação, os padrões obtidos na primeira são analisados.

Com isto, o algoritmo de partição usado pode ser considerado como outro tipo de análise de padrão local.

C Abordagens de Suporte ao Processo de Descoberta de Conhecimento

C.1 Ontologia como suporte ao processo de Descoberta de Conhecimento

Joseph Philips e Bruce G. Buchanan consideram que as ontologias podem ajudar no processo de descoberta de conhecimento, nos três seguintes aspetos: *"automatically suggest and generate new attributes based upon semantic and domain information, capture useful knowledge for reuse, and reduce the user's workload to interpret new tables"* (Phillips, et al., 2001). Para estes autores, uma ferramenta que seja capaz de armazenar a informação recolhida acerca de um determinado domínio, poderá ser útil para o KDD porque este é um processo iterativo, o que leva a que quanto mais informação for recolhida, melhor será o desempenho, e maior será a utilidade das ontologias (Phillips, et al., 2001).

Estes autores para além de quererem definir um processo útil que seja capaz de encontrar conhecimento *a priori* para um determinado domínio, também querem que este conhecimento possa ser usado noutros domínios, recorrendo apenas a uma ontologia.

Ontologia é uma especificação de uma conceptualização. Isto é, uma ontologia é uma descrição (como uma especificação formal de um programa) dos conceitos e relações que podem existir para um agente ou uma comunidade de agentes. Esta definição é consistente com o uso de ontologia como um conjunto de definição de conceitos, mas mais geral. E é certamente uma definição diferente da palavra que a filosofia lhe dava (Gruber, 1993).

A aplicação de ontologias na descoberta de conhecimento em base de dados é constituído por três componentes: o interpretador da tabela, um programa em *Prolog* e os dados de saída do *Prolog*.

O interpretador da tabela tem como objetivo inter**r**pretar um ficheiro de dados, reconhecer os atributos e os seus valores, inferindo assim atributos de domínio a partir dos valores e verifica o domínio questionando o utilizador. De seguida, relê o ficheiro inicial por forma a criar um programa *Prolog* que irá incorporar estes dados, sendo que este programa

contém anotações relativas aos atributos que indica como determinado atributo pode ser usado.

O programa *Prolog* é invocado numa segunda fase, usando uma ontologia existente e o novo programa criado, por forma a criar novos atributos conforme as regras definidas, que se baseiam nas anotações dos atributos criadas pelo interpretador. Este programa cria novos atributos e gera valores para esses atributos, escrevendo-os num novo ficheiro.

E por fim, o último componente tem como principal função produzir um novo ficheiro com a informação proveniente do ficheiro inicial e com os novos atributos e respetivos valores (Phillips, et al., 2001).

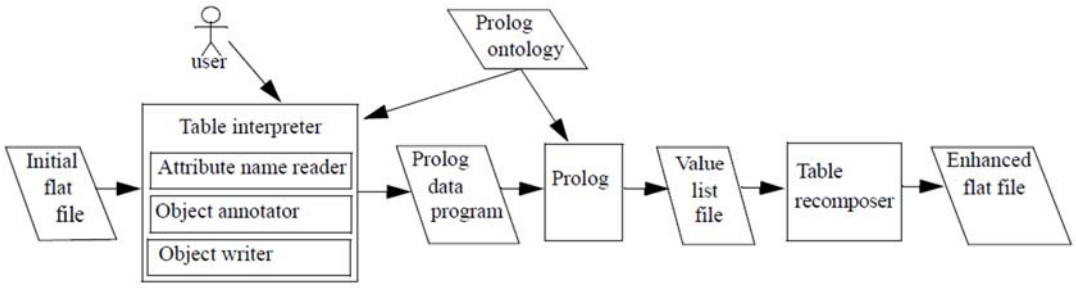

Figura 29 – Visão geral do processo (Phillips, et al., 2001)

C.2 Anotação Semântica e Serviços para Ferramentas de KDD de partilha e reutilização

A partilha de ferramentas de KDD através de uma ontologia específica para a conceptualização das atividades realizadas no domínio do KDD. Fornecendo uma conceptualização partilhada de recursos no domínio do KDD e das suas propriedades, essas ligações fornecem capacidades de fundação e de raciocínio semânticos (Semantic Annotation and Services For KDD Tools Sharing and Reuse, 2008).

Com recurso a serviços *Web*, é criada uma partilha de conhecimento e de ferramentas, o que permite reutilizar qualquer uma a qualquer momento, necessitando apenas de um acesso à Internet. Sendo o que conceito desta abordagem é a partilha de técnicas de KDD por forma a otimizar os processos e os algoritmos a serem seguidos, tendo em conta a informação

que é passada e que é validada junto da ontologia por forma a identificar o tipo de problema em causa e qual a abordagem mais correta.

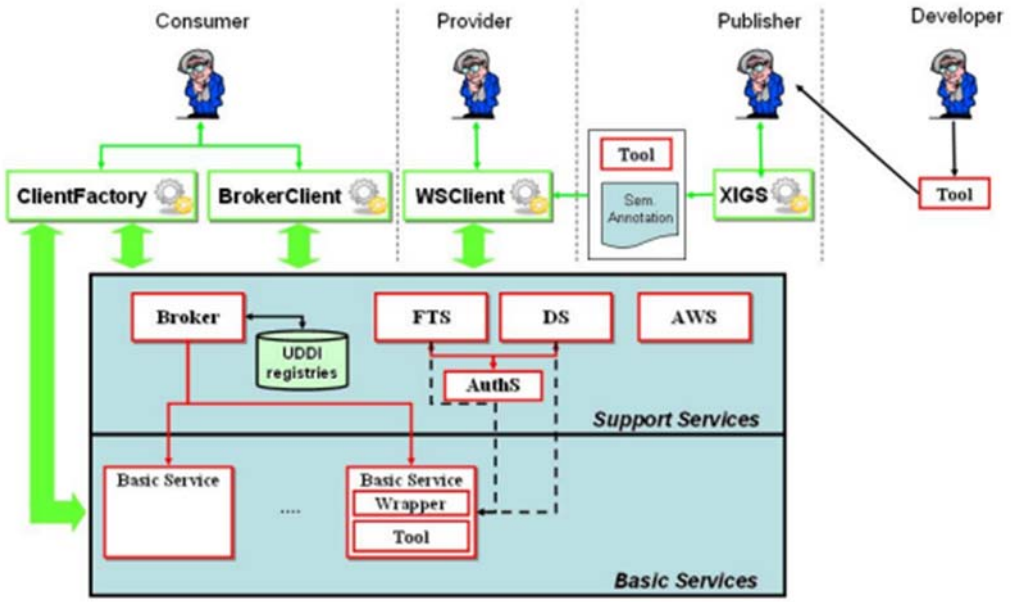

Figura 30 – Processo de funcionamento da partilha de serviços

D IKVM.NET

O IKVM.NET é uma ferramenta desenvolvida por Jeroen Frijters. Está é uma implementação do Java para o Mono e para o Microsoft .NET Framework. Esta ferramenta permite disponibilizar bibliotecas Java em ambientes .NET.

Esta ferramenta permite, entre outros: ter uma máquina virtual Java implementada em .NET; uma implementação .NET de bibliotecas Java, e disponibilizar ferramentas que permitem a interoperabilidade entre o Java e o .NET, permitindo o desenvolvimento de aplicações .NET em Java.

O IKVM.NET inclui o ikvmc no seu pacote de distribuição que permite a conversão dos ficheiros Java .Jar em bibliotecas .dll .NET e em aplicações .exe.

O processo de conversão é tão simples como abrir a linha de comando no Windows, navegar para o diretório onde se encontra o ficheiro ikvmc e executar o seguinte comando como este se encontra no Código 6.

```
Ikvmc [nome do ficheiro].jar
```

Código 6 – Execução do Ikvmc

Neste caso é necessário primeiro copiar o ficheiro que se pretende converter para o diretório onde se encontra o ikvmc. Após a execução com sucesso, basta voltar ao diretório e procurar um ficheiro com o mesmo nome, em que a única diferença será a extensão que esse terá (Frijters, 2010).

E Diagrama

E.1 Diagrama de sequência – WEKA

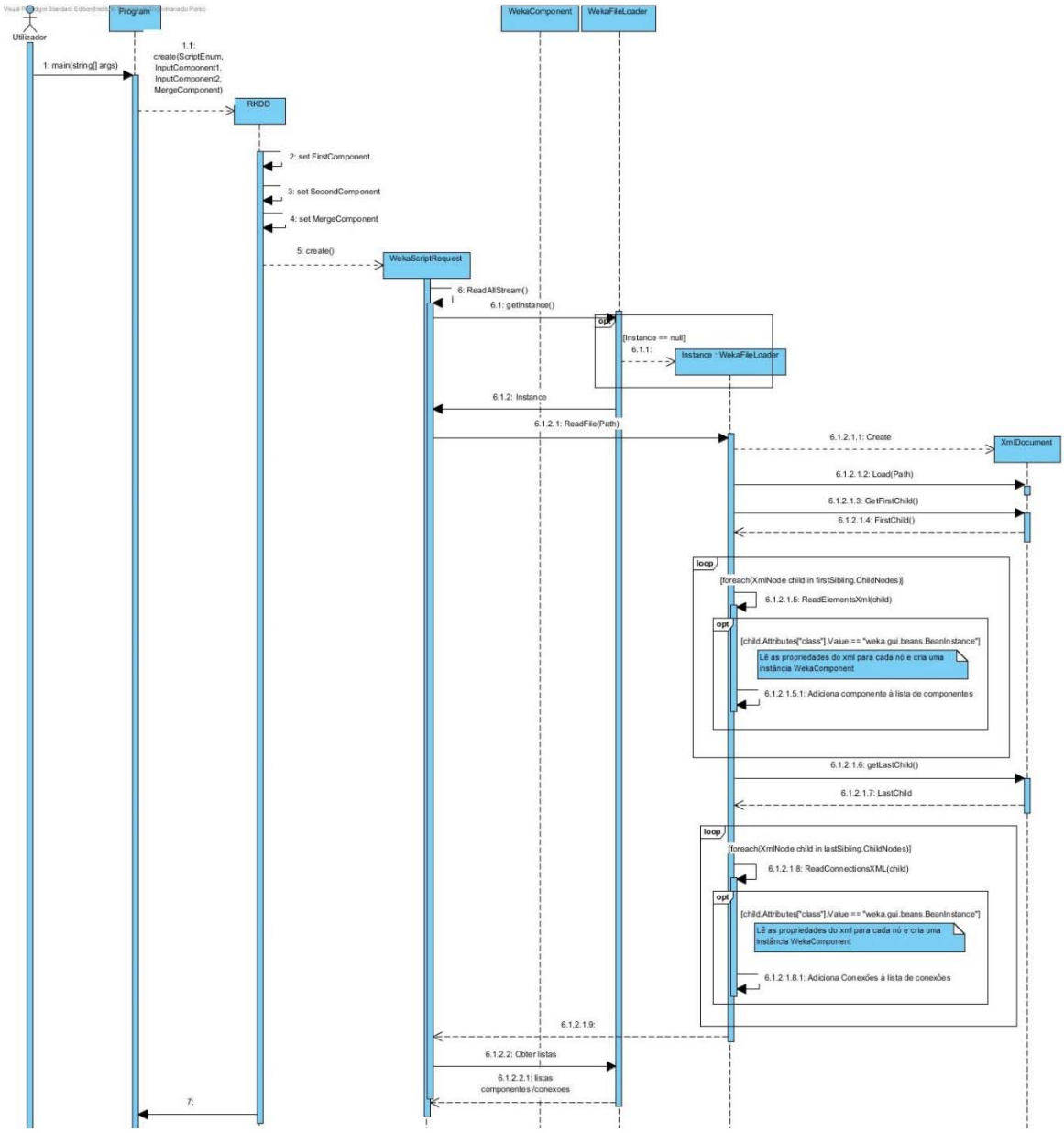

Figura 31 – Diagrama de sequência – implementação da comunicação com o WEKA (Parte I)

Nota: Na Figura 32 está representada a continuação da Figura 31 dada a dimensão do diagrama de sequência.

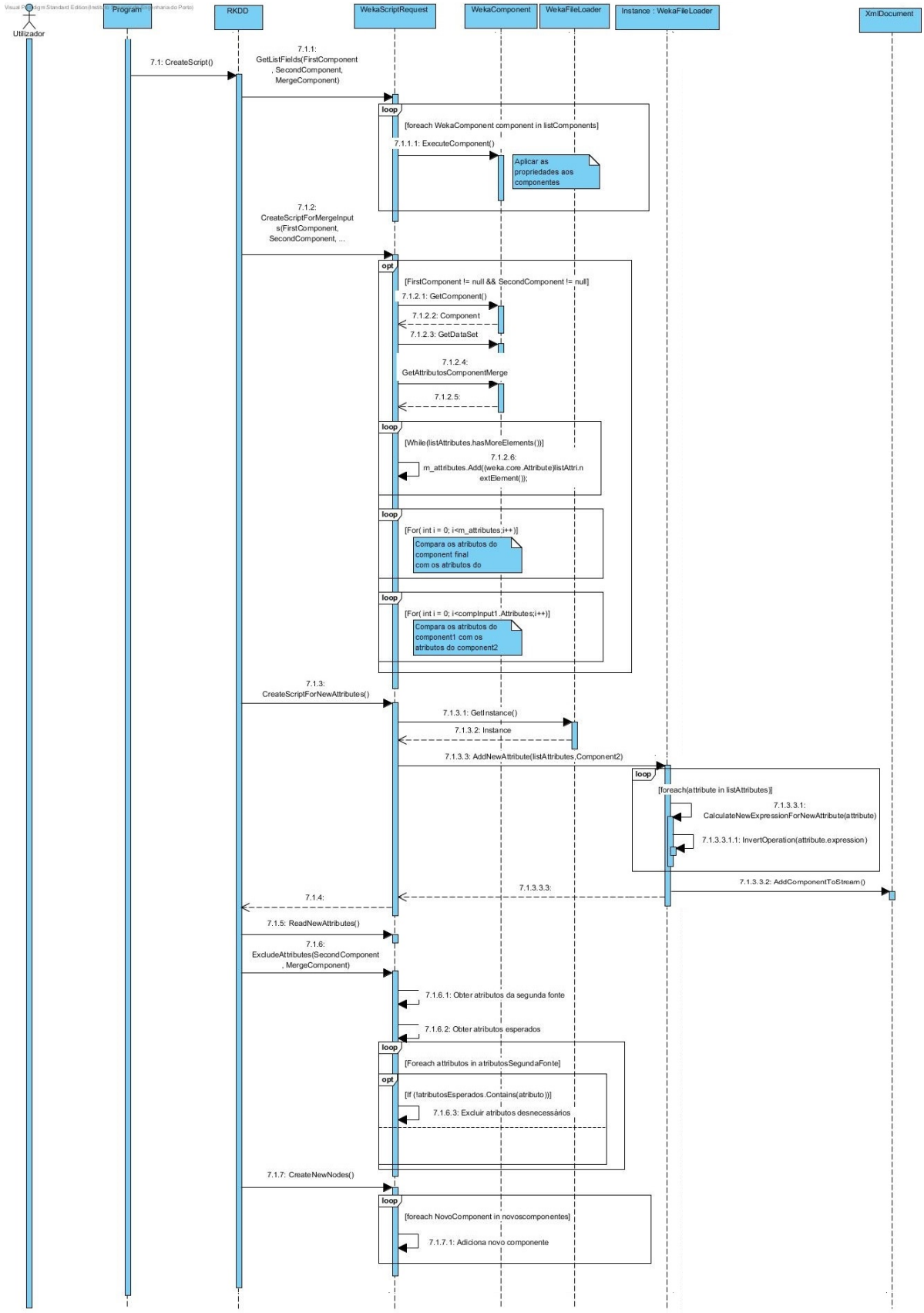
Utilizador
Program
RKDD
WekaScriptRequest
WekaComponent
WekaFileLoader
Instance : WekaFileLoader
XmlDocument
7.1: CreateScript()
7.1.1: GetListFields(FirstComponent, SecondComponent, MergeComponent)
loop
[foreach WekaComponent component in listComponents]
7.1.1.1: ExecuteComponent()
Aplicar as propriedades aos componentes
7.1.2: CreateScriptForMergeInputs(FirstComponent, SecondComponent, ...
opt
[FirstComponent != null && SecondComponent != null]
7.1.2.1: GetComponent()
7.1.2.2: Component
7.1.2.3: GetDataSet
7.1.2.4: GetAttributosComponentMerge
7.1.2.5:
loop
[While(listAttributes.hasMoreElements())]
7.1.2.6: m_attributes.Add((weka.core.Attribute)listAttri.nextElement());
loop
[For(int i = 0; i<m_attributes;i++)]
Compara os atributos do component final com os atributos do
loop
[For(int i = 0; i<compInput1 .Attributes;i++)]
Compara os atributos do component1 com os atributos do component2
7.1.3: CreateScriptForNewAttributes()
7.1.3.1: GetInstance()
7.1.3.2: Instance
7.1.3.3: AddNewAttribute(listAttributes,Component2)
loop
[foreach(attribute in listAttributes)]
7.1.3.3.1: CalculateNewExpressionForNewAttribute(attribute)
7.1.3.3.1.1: InvertOperation(attribute.expression)
7.1.3.3.2: AddComponentToStream()
7.1.3.3.3:
7.1.4:
7.1.5: ReadNewAttributes()
7.1.6: ExcludeAttributes(SecondComponent, MergeComponent)
7.1.6.1: Obter atributos da segunda fonte
7.1.6.2: Obter atributos esperados
loop
[Foreach attributos in atributosSegundaFonte]
opt
[If (!atributosEsperados.Contains(atributo))]
7.1.6.3: Excluir atributos desnecessários
7.1.7: CreateNewNodes()
loop
[foreach NovoComponent in novoscomponentes]
7.1.7.1: Adiciona novo componente

Figura 32 – Diagrama de sequência – implementação da comunicação com o WEKA (Parte II)

F Análise de valor

"Uma proposta de valor clara e bem definida é necessária para determinar o modelo de negócio" (traduzido de (MaRS, 2002)). Esta proposta deverá ser clara, na medida em que, é nesta que estarão as vantagens e o valor acrescentado do negócio, visto que sem existir evidências dessas vantagens é mais difícil de apresentar um modelo de negócio e justificar o porquê de ser determinado público-alvo o escolhido.

Sem uma proposta de valor é difícil ter a perceção de como é que serão geradas receitas, quais os parceiros que irá necessitar e quais as estratégias que serão seguidas para angariar clientes e como os fidelizar (MaRS, 2002). É com base numa proposta de valor bem definida que se consegue captar os potenciais clientes e fideliza-los através das vantagens apresentadas, evidenciando dessa forma os benefícios que este poderá ter ao estabelecer uma relação com quem desenvolve a solução. Para tal é necessário definir um valor que pode ser visto como necessidade, desejo, interesse, atitude, e preferência. Sendo que, por vezes, este valor é interpretado de forma diferente pelo comerciante e pelo cliente, denominando-se de valor percebido. Este consiste na utilidade que o consumidor dá ao produto ou serviço que está a adquirir, baseando-se nas perceções de como é recebido e de como é entregue, já que para ele o que lhe interessa é que perceba que o valor do negócio é justo e que satisfaça as suas necessidades, conseguindo estabelecer uma relação com o comerciante.

Desta forma é necessário que seja criada uma proposta de valor que ilustre o que o comerciante/vendedor tem para oferecer ao cliente, e qual o valor que o cliente adquire na aquisição de um produto/serviço.

No caso da solução a ser criada, esta destina-se a clientes que pretendam obter resultados oriundos da extração de dados de uma forma mais rápida e em que possuam esses mesmos dados espalhados por diferentes fontes com um esquema de dados distintos, o que exige sempre um maior sacrifício do analista na junção dos mesmos devido ao facto de ter de configurar estes de acordo com um esquema que é esperado.

Esta solução permitirá reduzir o tempo de configuração e mapeamento de diferentes fontes, o que levará a que o analista se consiga focar mais na informação que conseguirá extrair dos dados e não como é que este processo de mapeamento é executado.

A redução do tempo de configuração é um dos benefícios, sendo que os restantes benefícios advêm deste que será o principal. O facto de ter a informação agrupada e até poder ser extraída como um todo num processo de migração, faz com que seja uma mais-valia, visto que pode ser dada outra finalidade para além da extração de dados, ou a realização do processo de mineração de dados sobre esses mesmos dados agrupados.

Desta forma, é uma solução que irá simplificar o processo de extração de conhecimento do mesmo domínio, apenas aplicando regras de mapeamento entre as fontes de dados e acrescentando uma regra implícita na estrutura que deve ser seguida aquando do uso da mesma.

Quando se define uma proposta de valor e os benefícios do que se pretende colocar no mercado, o cenário de negócio a ser seguido deverá ser definido antecipadamente por forma a clarificar na negociação o que se pretende.

Desse modo, o cenário que seria adotado era o *Win-Win*, ou seja, ambas as partes sairiam beneficiadas com o negócio. Mas, para que este cenário seja cumprido, existem alguns procedimentos necessários que ambos deverão ter em conta, como: definir claramente as expectativas e objetivos do negócio a ser estabelecido; identificar quais os pontos que são indiscutíveis; prever as contraofertas que poderão existir durante o processo negocial; ter pleno conhecimento de todos os aspetos que o negócio poderá envolver; identificar quais as necessidades e desejos da outra parte; estabelecer limites entre o que dar ou receber; assim como explicar o porquê desses limites estabelecidos.

Com este tipo de negócio todos ficariam a ganhar, já que permite uma reutilização de um processo, e desta forma, um maior reaproveitamento do tempo para outras tarefas.

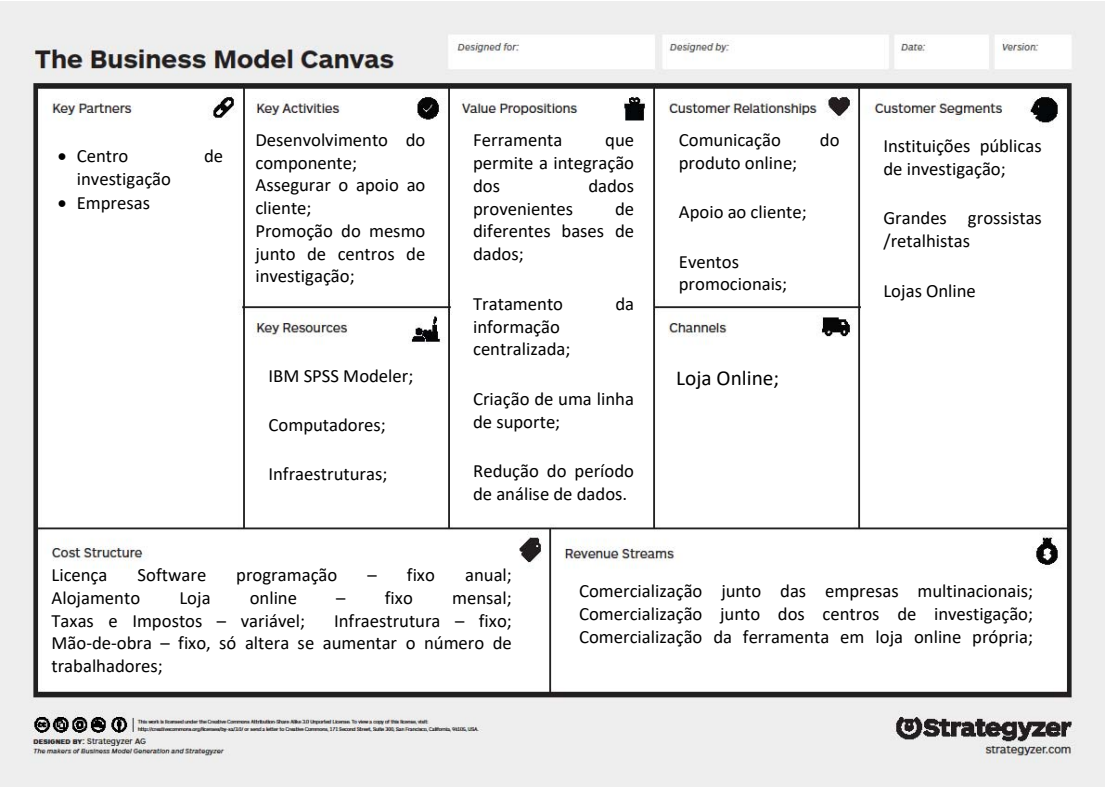

Figura 33 - Modelo de negócio de Canvas

Na Figura 33 está definido o modelo de negócio Canvas que irá ser a base para o desenvolvimento e comercialização da solução.

Como parceiros estão identificados os centros de investigações e as empresas, sendo que estas são duas das áreas que possuem para o mesmo tipo de dados uma variedade de fontes com esquemas de dados diferentes o que permite que o seu modelo seja usado como um verdadeiro teste à solução desenvolvida. Sendo que a posição destas no mercado e a credibilidade que possuem é um ponto de partida para que publicitem a solução e que a façam chegar a mais instituições.

Como atividades principais: o desenvolvimento da solução, a promoção da mesma e o suporte aos clientes, uma vez que poderão sugerir ou adquirir ajustes.

Para assegurar o desenvolvimento é necessário recorrer ao *software* da IBM, o *SPSS Modeler* que é uma das grandes referências a nível mundial na área da mineração de dados, assim como o equipamento para desenvolver a solução desenhada.

Esta solução traz ventagens para as instituições e organizações, na medida em que permite a longo prazo uma redução no período de tempo que é despendido pelo analista na

junção de várias fontes de dados, permitindo dessa forma agilizar o processo de centralização da informação.

A divulgação do produto via internet será um fator decisivo, pois este será mais fácil de divulgar junto de entidades internacionais. Uma vez que estas se encontram dispersas por vários pontos do planeta e em que é dado a conhecer ao Mundo esta solução, mas não dispensa de uma futura relação próxima com o cliente, dando o apoio e suporte necessário.

Como já referido anteriormente, este será um produto para instituições ou organizações com um volume de mercado mais acentuado em que possuem fontes de dados dispersas e necessitem de uma ferramenta que lhes poupe tempo na junção das mesmas para produzir as análises que pretendem.

Para que esta solução seja criada, é necessário definir quais os custos que estão associados, sendo que, como custos fixos existem as licenças de *software*, o alojamento da loja online, a mão-de-obra, e as infraestruturas. Por sua vez, as taxas e impostos são custos variáveis devido à faturação.

Para a criação de valor existe um método denominado de *Analytic hierarchy process* (AHP) que perante uma decisão a ser tomada obriga que sejam estabelecidos critérios, subcritérios, por forma a optar pela melhor alternativa; para isso, são atribuídos pesos a cada critério, são definidos entre eles qual o mais importante.

No entanto, este é um método de avaliação que deve ser utilizado quando se possui 3 ou mais aplicações que se possam comparar, assim como 3 ou mais critérios de comparação. Tendo isto em conta, este método não se pode aplicar, visto que o número de aplicações identificadas é inferior a 3 e como tal, a avaliação do valor da solução será avaliado com o decorrer do tempo de utilização. O valor percebido que este projeto tem é um valor racional derivado do consumo por parte dos utilizadores que pretendam analisar este tipo de informação (Conceptualising 'Value for the Customer': An Attributional, Structural and Dispositional Analysis, 2003) . O valor percebido deriva do consumo[16] e da utilização que este terá. Desta forma, este irá variar conforme a utilização e a importância que representar para os seus utilizadores.

[16] Deriva do consumo – traduzido do inglês derive from consumption