UNIVERSITY OF GOTHENBURG

Designing a Decision Support System with Incorporated Data Mining

A Software Design Project

Bachelor of Science Thesis in the Programme Software Engineering & Management

CARL BERGLUND

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
Göteborg, Sweden, June 2015

Designing a Decision Support System with Incorporated Data Mining}
A Software Design project

CARL BERGLUND

Examiner: Michel R. V. Chaudron

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden June 2015

Designing a Decision Support System with Incorporated Data Mining

Carl Berglund
University of Gothenburg,
gusbercaal@student.gu.se

Abstract— **In this paper we propose a software design for a Decision Support System (DSS) with incorporated Data Mining (DM), which integrates data mining operations such as classification and clustering. With the increasing amount of data available today, decision support benefits greatly from a stronger data foundation. We introduce a business problem and propose a simple design solution and evaluate it against a set of well defined requirements to assess its potential to provide decision support. The simplistic design proposed in this paper is a new take on decision support in the fiber market.**

I. Introduction

The aim of this project was to design and evaluate a Decision Support System (DSS) that will aid the planning of future fiber infrastructure. A company (hereby referred to as “client”) is looking to make their process of deploying fiber technology more efficient and accurate. In order to provide valuable decision support for client in the planning process and marketing activities an automated system with incorporated data mining (DM) was designed. The design embeds a template profitability scoring model based on data mining operations such as classification. Furthermore, clustering activities were employed to identify and return groups of customers with a high profitability grading.

A. Case Company Description

Client is a large administrator of fiber and cable networks in Sweden and plan projects to deploy “fiber areas” where suitable. In a fiber area, households are offered the installation of fiber technology at a flat rate, where they would otherwise pay for installation based on the household adjacency to nearest connecting point.

Client administrates a large amount of cable in the soon-to-be outdated cable network. Cable is expensive to maintain and is outrun by fiber and wireless technologies, thus the client is interested in replacing copper cables with fiber technology. However, deploying fiber is costly, and projects should be planned carefully to ensure efficiency. Carefully, in this case, means ensuring that residents in a given area are likely to upgrade to fiber if offered. By considering factors such as age, average income, current technology, other technologies, and location, we can estimate how profitable (probable to purchase) a customer/household is. By designing a software solution to yield a probabilistic model of profitability, and render an area with a high density of profitable customers, the client would have a stronger basis for their marketing strategies and, consequently, their deployment projects.

The client have a long standing relationship with Purple Scout AB, a software consultancy company located in Gothenburg, with which this project was undertaken. Purple Scout maintains a larger Business Intelligence (BI)-suite with numerous applications for the client, on which our proposed solution was based.

B. Data Mining

The traditional method of extracting knowledge out of data sets usually means manually inspecting and analyzing information. With the rapidly growing volumes of data available, new tools are constantly in demand to make sense out of the data. This problem is the basis for the emergence of data mining. Data mining is concerned with making sense of data in large volumes, finding patterns and discovering information in data sets that are too big, or too low level, or too complex for humans to digest [1]. Data mining is a crucial next-step in the world of data, as the volumes of data grows and the data itself increases in complexity. Intelligently analyzed data becomes a valuable resource that can lead to new understanding and competitive advantages. Data mining is a general term for this kind of information discovery in data sets, and consists of a multitude of methods. A few of those methods are [2][3]:

Classification A predictive learning function that classifies a data item into one of several predefined classes. Classifications are discrete and are used to determine a nominal target for an instance. A predictive model with a numerical target uses regression instead.

Regression A predictive learning function that maps a data item to a real-value prediction variable. Regression is used to predict a numeric value based on other attributes as well as on historical data.

Clustering A common task in which one tries to identify a set of categories to describe the data. Clustering is useful when there is no existing class to be predicted but the instances need to be divided into natural groups.

Summarizations A descriptive task that involves methods for finding a description of a data set.

Dependency Modeling Finding a model that describes the dependencies between data or data sets.

Change and Deviation Detection For discovering significant changes in data sets.

DM is often mentioned together with machine learning, and vice versa, and definitions may vary. Simplified, one could say that machine learning is a collective name for the tools, algorithms, used in DM application [3]. With the different methodologies of DM listed above comes many different machine learning schemes. However, no machine learning scheme is applicable to all data mining problems [3].

C. Decision Support System

Decision Support System (DSS), is a general term and does not restrict itself to any particular application or industry. As the name suggests, is a system that supports decision making but that definition allows the inclusion of a lot of different applications. Further refined, a DSS is a computer-mediated system that provides information to aid users in their decision making in scenarios that are long- or short-term, strategical or tactical, and the requisites behind it can be of various degrees of complexity [4][5]. DDS is usually considered a cornerstone in BI (Business Intelligence), and the systems typically assist in managerial decision making [6]. A simple application of a DSS can be a recipes website, where you filter your results based on preference, allergies, or how many people you are serving and for what occasion. More complex applications can include systems that allows insight in the progress, costs, inventory, and delays in constructions projects, as in [5].

Classic DSS techniques include sophisticated data management capabilities, model management systems, simple but powerful user interfaces that allow interaction with databases, report generation and analysis tools [4] and was best defined by Gorry and Scott [7] in 1971 [4]. Since then DSS have evolved to include several new concepts, but the four most significant contributions to DSS system are the techniques of data warehousing, online analytical processing (OLAP), data mining and the web-based DSS [4]. In [5] a DSS with support of a data warehouse is described. The data warehouse is tasked with ensuring the appropriate data is available at the right time to the right end. A data warehouse can store raw data, preprocessed data or both, and allows for powerful querying of data, such as historical, previous or prejected data. Together with OLAP this enable some powerful data extraction, used to answer complex questions such as "What are the supply patterns for construction projects of type x?", or "What is the projected supply demand for project y?". Generally, data warehousing is about centralizing processed or unprocessed data to one domain, reducing the processing and management efforts and costs [5].

With the use of DM a DSS can benefit from making sense of larger amounts of more complex data to generate strong decision support. A powerful application of DM in decision support could be a system that governs a bank's lending policy by judging a loan recipient based on banking history and credit score, as proposed in [8]. While regarding the banking industry, their concerns are the same as for any industry, as are their motivations. A DSS is motivated by the banks need to make good decisions to minimize risk associated with the companys activities [8].

This paper contains 5 sections as following: This section has introduced the reader to the case company, data mining and decision support systems. Section II, Methodology, will outline how we developed our design. Section III, Results, will oresent our findings and motivate the proposed design. We then discuss the results in Section IV, and lastly we draw conclusions from this project in Section V.

D. Purpose

This DSS was developed to provide the client with tools to gain stronger informational basis for their future deployment projects, as well as their marketing focus. The research contribution of this paper is the process of designing a DSS with incorporated DM to be deployed on an existing system, with the tailored requirements that come with it. This paper will present the reader with the process of development and the evaluation of a software design that serves as a proof of concept for decision support in the fiber market. The focus will be on how DM technology is applied in software design deployed on an existing system. Further, this paper will discuss the design, its development process, quality, and social utility.

RQ How can we design a Decision Support System with incorporated data mining?

II. Methodology

This design assignment was conducted for, and with, Purple Scout AB. With considerable experience with Open Source Software, Purple Scout offers a data mining-based approach for solving the clients problem, using software embedded in an Open Source business intelligence suite called Pentaho [9].

This design assignment was motivated by the need to employ an IT artifact to solve the clients problem, in a field that to our knowledge had not seen it utilized before. [10] defines six steps to design science that was deemed a natural course of action in this exploratory learning-centered endeavorer. The steps are; Identification of problem, Suggested solution, Development, Evaluation, Results, and Impact. This paper will be proceed with the following structure: Identification of problem, Suggested solution, Development, and Evaluation.

Results is then presented in Section III, and Impact in Section IV.

A. Identification of Problem

The client expressed their concerns about the process of zeroing in on a particular area of operations. Their concerns regarded the process not being satisfyingly intelligent, nor was it efficient or easy to evaluate. Their main concern was that projects were planned based on some polling and qualified guesses of interest in areas that were investigated.

B. Suggested solution

As suggested by Purple Scout, the artifact is a DSS that performs data mining operations on data supplied by client. Initially, we knew very little about DM and how to use it, and so the suggested solution would come to change many times over the course of this project. Once the problem was identified, the client was not involved much in the process. Our solution would change, but the problem remained the same. Considering requirements for the software, the representative from client had very little to say regarding technology. Purple Scout offered their idea using data mining and the motivation for this approach originated both from their perception of data mining as a sound approach to a data heavy problem, but also from their want to learn more about data mining. The software in mind was Weka, a data mining framework in the PentaHo suite.

Further, we decided to design a component that could be deployed in Purple Scouts existing BI-system, a Business Intelligence platform developed and maintained for other business needs of the client. Therefore, we determined that the DM component should be a black box component that should impose no dependencies on Purple Scout's system (hereby referred to as PSBI for Purple Scout BI System). By developing our artifact on their existing architecture we would save considerable amount of time, but it would also mean some constraints to our design. The main constraint to the design were programming language, external packages, and interfaces to the PSBI. With external packages and interfaces we mean that we could use external packages, but the interface must of of a data type that PSBI can accept.

C. Development

Weka is a state-of-the art machine learning workbench developed at the University of Waikato, New Zealand [3] and the name stands for Waikato Environment for Knowledge Analysis. It is written in Java and provides two tools important for this project; the Weka Explorer, and the Weka library. In Weka explorer, a user can get hands on with pre-processing tools as well as machine learning schemes. In the Weka explorer the user can pre-process data, apply a machine learning scheme, and analyze the data without writing any code at all. The Weka library is a .jar file that can easily be included in any Java application and contains all the tools available in the explorer. The explorer would prove very useful in getting familiar with Weka, and how and when to apply which algorithm. With each pre-processing filter comes numerous settings than can be applied, and same goes for each machine learning algorithm.

Once we knew what we could do with Weka, and how to do it, we transferred our knowledge to our software design. We created an Software Requirements Specification (SRS) to establish a general direction for our design, and by considering our vision and our restrictions, we developed requirements. The client delivered the problem rather than the solution, and was thus part of defining the output of the system, but less of the how-to. Together with Purple Scout, we reasoned about how the system should work, yielding requirements. At a later stage of the project we produced user stories with acceptance criterias in order to properly evaluate the design. These stories and criterias were written to be tightly coupled with the requirements in order to evaluate of the requirements based on fulfilled acceptance criterias. As our system have no explicit user interaction, we would consider PSBI our user. The SRS was refined many times over during the steps mentioned in this section. The evaluation of the artifact would result in renewed requirements, and discoveries by prototyping would impact how we perceived the system. A typical process for carving out the requirements would be to discuss what we wanted to do, to take on a partial problem and discuss what we needed to do to solve it. By studying methods of data mining and testing theories in Weka Explorer we would find a suitable solution and refine that solution into a requirement.

Parallel to the design development we produced some code. While our goal was not to deliver a runnable software, testing our theories in code helped us understand DM better, as well as it provided us with instant feedback that would impact our design. As DM was a very new to us, our research was very much exploratory, and rather than having the design as basis for our code production, it was the other way around for most of the time. Trying, testing and evaluating code executions played a part in exploring what machine learning can be used for, and how to interpret the results.

D. Evaluation

[11] states that quality, utility, and efficacy of the design are three requirements that must be evaluated in order to assess rigorousness. "A design artifact is complete and effective when it satisfies the requirements and constraints of the problem it was meant to solve" [11]. With this quote in mind, we verify the solutions efficacy, quality and its utility based on the designs ability to solve the problem. We evaluated the artifact based on the conformity of the design with its requirements - the quality notions that the design must be proven to fulfill. To yield tangiable argument to a requirements fulfillment, we added acceptance criteria with hard requirements. Each requirement is related to one or several acceptance criteria of a user story, and would together

build a stronger confirmation of its validity. We used the descriptive evaluation method as proposed by [11] to build a convincing reason for the fulfillment of each acceptance criteria in order to give substance to the argument of a requirement's fulfillment.

Furthermore, we performed a design review, an informal peer-based review of the design as suggested in [12]. We produced a checklist, based on a template from the University of Minnesota [13] that was altered to be used in our scope. The design was presented and explained to a software developer from Purple Scout who would then part-take in a walk-through, a review session where we explained the flow of the system and the attendant evaluated the design based on bullets in the checklist. The checklist, attached in the appendix of this paper, is a guide for evaluating the design and its coherence with the requirements, ranging from naming convention and syntactical correctness to design completion. Another purpose of this review was to allow the attendant to trace each requirement through acceptance criteria to design and point out potential weaknesses and question marks and ultimately pass or fail its fulfillment.

III. Proposed Design

This section will explain the design, explore the requirements that led to this solution and present the reader with finalized design in diagrams. The design will then be explained and motivated.

1) Concept: The proposed system serves the purpose of generating a cluster representing an area with a high density of highly profitable customers. To do this, we needed to identify high profitability customer and produce a natural grouping of profitability. A customer will be ranked based on following attributes; income, current connectivity, and age.

We would leave the exact values and scoring scheme to the relevant business managers at client, but one can argue that a customer that receives a high rating is a 30 year old person with high income, currently connected to older cable protocols, and currently uses IP-TV, while a customer that receives a lower rating is over 65, with low income and no current uplink at all.

When high profitability customers are identified, the system will return a list of clusters with high profitability customers. A cluster is a natural grouping of elements based on mean values of selected attributes [3]. In our case, we want to find natural groups of customers with a high profitability grade, based on their geographic location, their x/y-coordinates.

2) Requirements: In the out forming of an SRS we established requirements that would have to be fulfilled in order to accept the design. The requirements are as follows;

REQ-1 The system shall be able to load a dataset with customers.

REQ-2 The system shall apply a scoring model to establish profiles of high to low profitability in customers.

REQ-3 The system shall perform a classification task on a dataset to determine each customers profitability.

REQ-4 The system shall output the classification results in a new class attribute.

REQ-5 The system shall perform clustering operations on dataset.

REQ-6 The system shall return data to its caller as clusters of customers.

REQ-7 The system must not impose any commercial licenses on PSBI.

3) Acceptance Criteria: Below are criteria that must be accepted for the design to be considered validated. Within square brackets, such as [REQ-1] is the relevant requirement that the criteria relates to.

AC-1 [REQ-1] A customer is of type Java Object and as specified in the interface from PSBI.

AC-2 [REQ-2] The scoring model is applied to a set percentage of the entire dataset.

AC-3 [REQ-2] The scoring model is only used if scores are not present in the dataset.

AC-4 [REQ-3] Classification is called using a subset of the data as training set.

AC-5 [REQ-4] A new class attribute *profitability* is created for all entries in the dataset. This class represents profitability and range between A - J, where A is the most profitable.

AC-6 [REQ-5] The clusters are generated from customers with different *profitability* ranking.

AC-7 [REQ-5] The clustering is based on a customers x/y-coordinates and its *profitability* class.

AC-8 [REQ-6] The clusters are returned in a generic data type.

AC-9 [REQ-7] The system only uses software that complies with the GNU General Public License.

4) Design: A generalization of the system as shown in Figure 1 provides an overview of the design. From this view the system is of course largely abstracted, so what can it tell us? What we can see are the two interfaces that the system

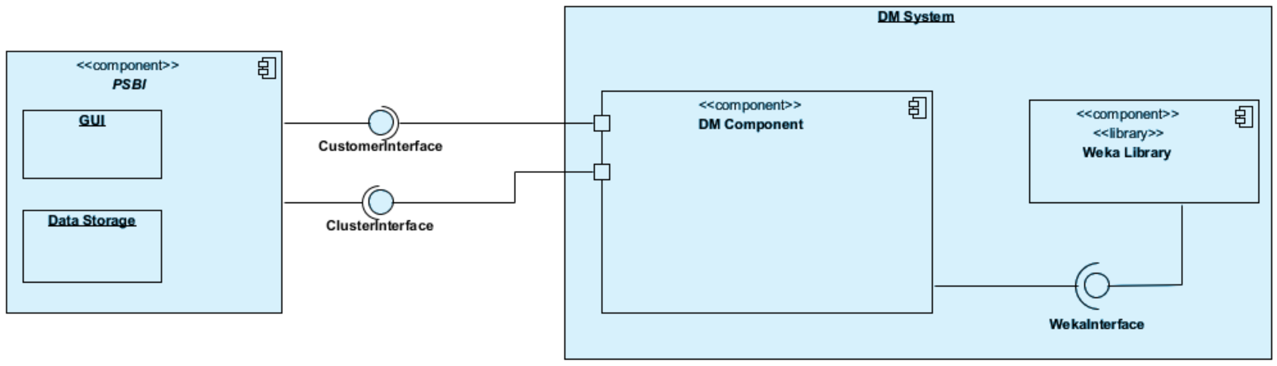


Fig. 1. UML: Component Diagram of PSBI and the proposed system

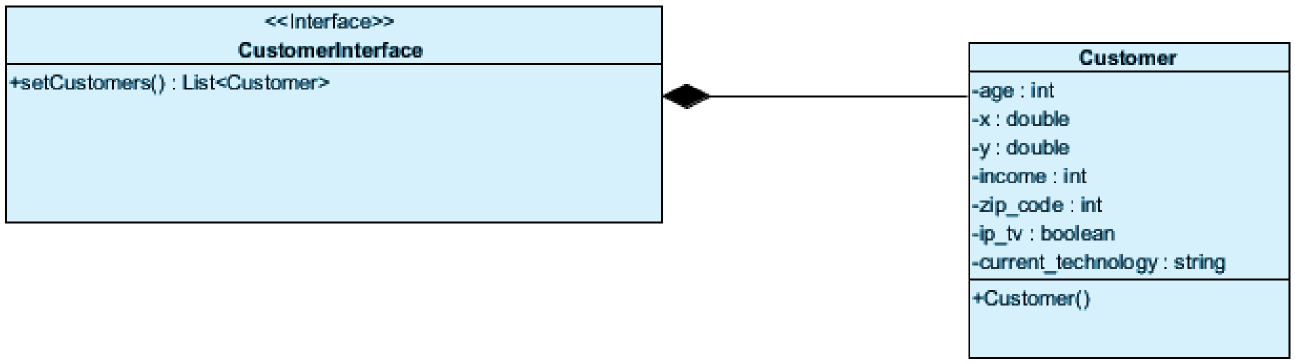


Fig. 2. UML: CustomerInterface and Customer class

uses to interface with PSBI. The CustomerInterface, which is the input of the system and detailed further in Figure 2, and ClusterInterface.

The ClusterInterface is the output generated by the DM System, as seen in Figure 3 on page 8. The output of our system holds attribute *profitability*, which makes out the basis for each cluster, and is not available in the (from PSBI) provided class Customer. By extending Customer to ClassifiedCustomer we wrap the new attribute in a wrapper that is recognized by PSBI. The Cluster is a generic typed list and contains ClassifiedCustomers. The motivation for the ClassifiedCustomer is as mentioned the need to provide a *profitability* attribute, this is done with Enum Profitability.

The classes that make out the system are modeled in Figure 4 on page 9. Weka compromised the object oriented solution for their own interface Instance on grounds of gained performance [14]. The class DataParser is thus needed to convert the list of Customers to a dataset Instances.

The class Classifier is used to assign profitability attributes. A training set must be supplied, which is based on a subset of the dataset with each instance's profitability assigned already. The reason for this is that while a classifier is used to predict an instance's class, the classifier need a model to apply to the set. With model, we refer to a set of rules built upon another dataset that have that class assigned. In this case, the model must be based on a dataset of customers with a set profitability class, and then apply that model to the test set. This division of the data is done in PreProcessing, where the dataset is first split into training and test set by percentageSplit and returned as an array with two sets of Instances. We apply the scoring model to the first subset in applyScoringModel which takes data

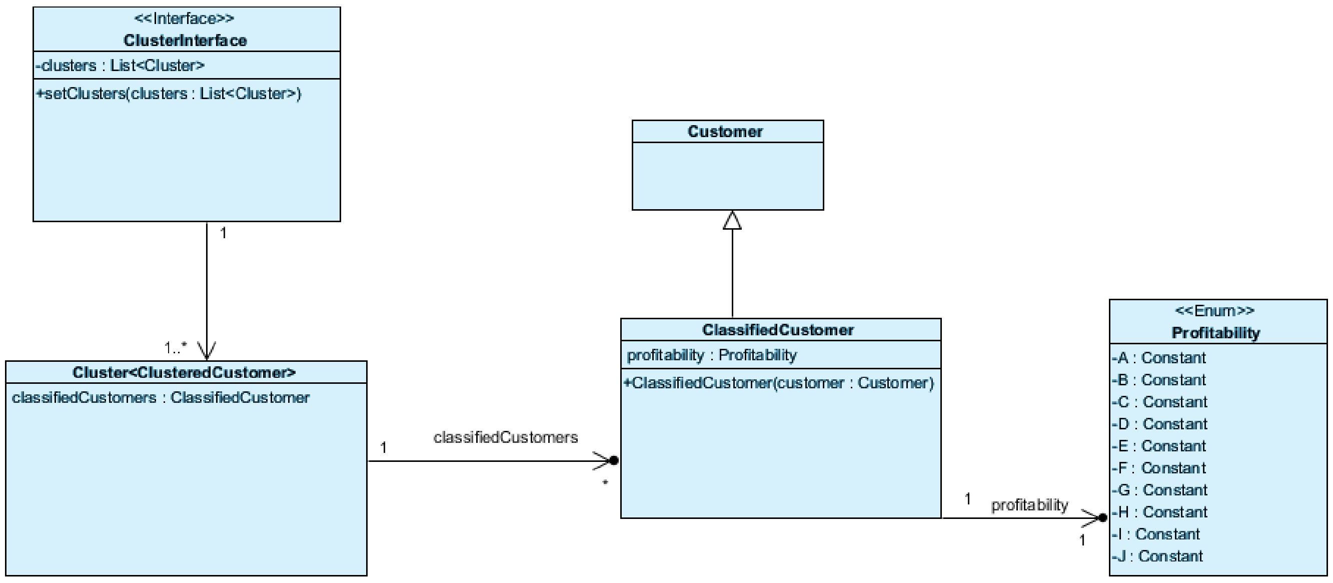


Fig. 3. UML: ClusterInterface interface

type Instances as its input. For each Instance in that set of Instances, a score can be calculated on the instance's attributes via getScore. The returned score is then used with getProfitability and mapped to a Profitability Enum using the map scoreToProfitability.

The scoring model is defined in this paper as a necessity for the classification tasks. However, a scoring model as such would be established by business managers or other professionals with insight of what parameters are in play when determining a customer's profitability. A scoring model could work like the following example;

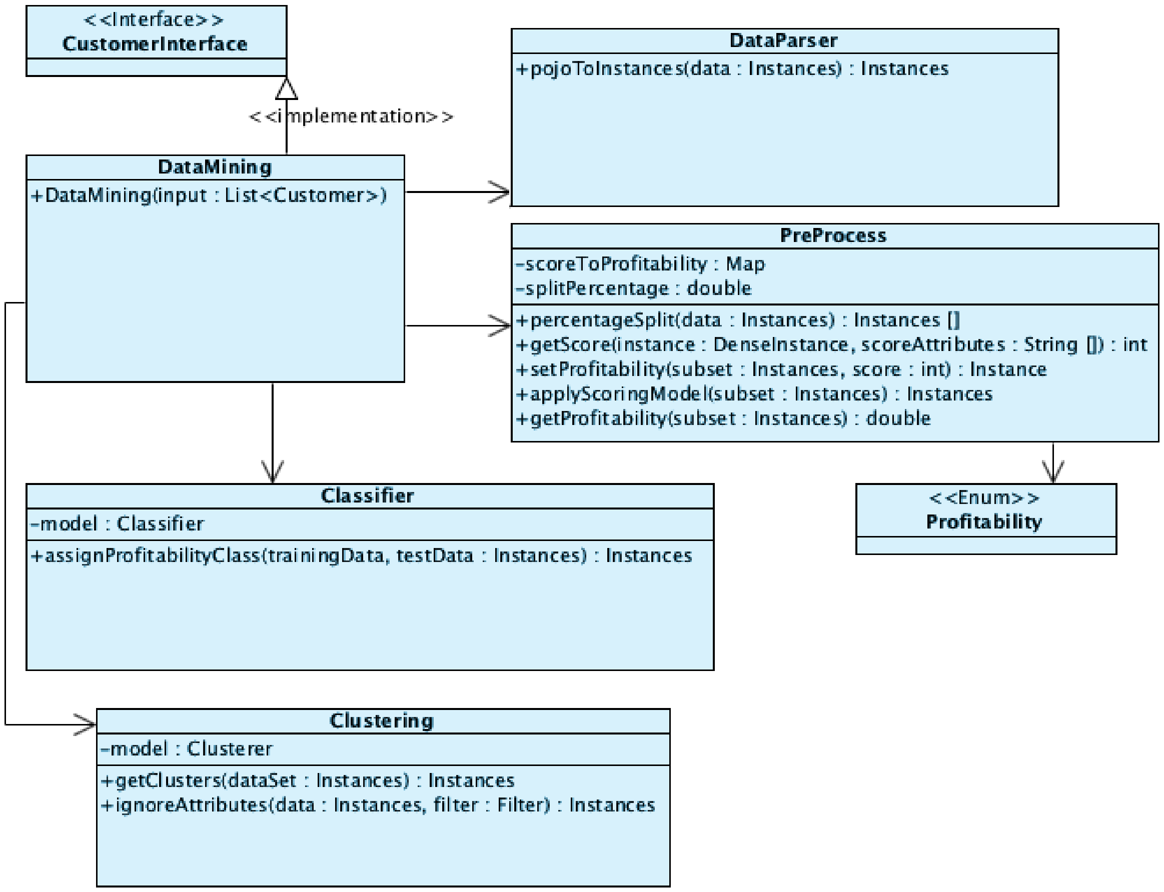
- age, lower is better. Based on the assumption that younger people are more interested in stronger technology for browsing, streaming, online gaming, etc.
- income, higher is better.
- current technology, where lesser technology is better, but no uplink is worse than fiber. Based on the assumption that a person with lesser technology and lower speed is more likely to upgrade than some with a lesser technology but with high speed. For example, a customer with cable technology that gets 24 Mbit/s(megabit per second) is assumed to be more likely to upgrade than someone who also is connected through cable, but get 200 Mbit/s.

So why not do this on the entire dataset? The assumption that the model is bulkier and less performance effective than a proven machine learning algorithm is basis for this approach. To keep things swift, the subset should not be too large, but a subset too small could yield insufficient experience for the model, where it has not encountered enough instances to build a reliable model. Before the model is called however, a check is performed with the method *getProfitability* which checks if the subset already has the profitability attribute set and returns a double representing the percentage of the set that owns said attribute. If that percentage is larger than field splitAttribute in PreProcess, the model is not called since there is already a model for profitability within the dataset, and the execution continues with classification.

As an alternative to building this primitive scoring model, one could argue that clustering is the logical choice as it is designed for finding natural groups in data. However, clustering, as we know it, finds natural groups based on mean values of instances attributes. Simply put, this would not generate ten distinct groups of profitability as we want, but rather produce ten natural groups found in the set. Thus we build our own model, however, it is possible that this can be done with other clustering schemes, but if that is the case it is beyond our knowledge.

Further, the classification operation is performed through the function call assignProfitabilityClass in the Classifier class. This method takes two arguments of type Instances, which are our trainingData, and dataSet[1], the other return from percentageSplit. In this method, every Instance is classified using a Classification scheme J48 and a profitability attribute is set. J48 is an algorithm used to generate a decision tree on classification tasks[3]. The returned data is a set of classified Instances.

Considering clustering - as we want to base our clustering



<<Interface>>
CustomerInterface
<<implementation>>
DataMining
+DataMining(input : List<Customer>)
DataParser
+pojoToInstances(data : Instances) : Instances
PreProcess
-scoreToProfitability : Map
-splitPercentage : double
+percentageSplit(data : Instances) : Instances []
+getScore(instance : DenseInstance, scoreAttributes : String []) : int
+setProfitability(subset : Instances, score : int) : Instance
+applyScoringModel(subset : Instances) : Instances
+getProfitability(subset : Instances) : double
<<Enum>>
Profitability
Classifier
-model : Classifier
+assignProfitabilityClass(trainingData, testData : Instances) : Instances
Clustering
-model : Clusterer
+getClusters(dataSet : Instances) : Instances
+ignoreAttributes(data : Instances, filter : Filter) : Instances

Fig. 4. UML: Class diagram of the DM component

operations on profitability and x/y-coordinates, we filter out the other attributes with ignoreAttributes which takes a Filter as an argument. This Filter is provided in the class and specifies the attributes that need be filtered out. Note that the attributes are still in the set, but invisible to the clustering methods. The returned filteredData is then the argument for getCluster method from the Clustering class. In getCluster, a clustering algorithm called simpleKMeans is used to group the Customer instances based on their filtered attributes. This is also where the instances are made into Customer objects again, but Customer with the profitability set, namely ClassifiedCustomers. Each ClassifiedCustomer is created within respective cluster, and the clusters are returned to DataMining where it is made available to interface ClusterInterface.

IV. Results

The results of this article is the finalized artifact and its potential to answer the research question. At a Minimum Viable Product (MVP), the design does fulfill this criteria. The research contribution is the artifact and the considerations that led to its completion. To add quality to that contribution, further research will be recommended in its respective section. When the design passes our evaluation efforts, we consider it a suitable design for a MVP.

A. Design Decisions

An important design decision that emerged from our increasing understanding of DM was the removal of the user interaction. Initially, we intended to also develop our own user interface with controls that would govern the machine learning algorithms at play. Over the course of the project, the user came to play a smaller role in the system, and was ultimately removed from our component. Our reasoning behind this decision was the gained understanding of the complexity behind the operations performed with use of the Weka library. To allow satisfying interaction with the system, a user would be expected to truly understand the data, the algorithms, and the output. To be able to use the system with ease the user would need considerable proficiency with

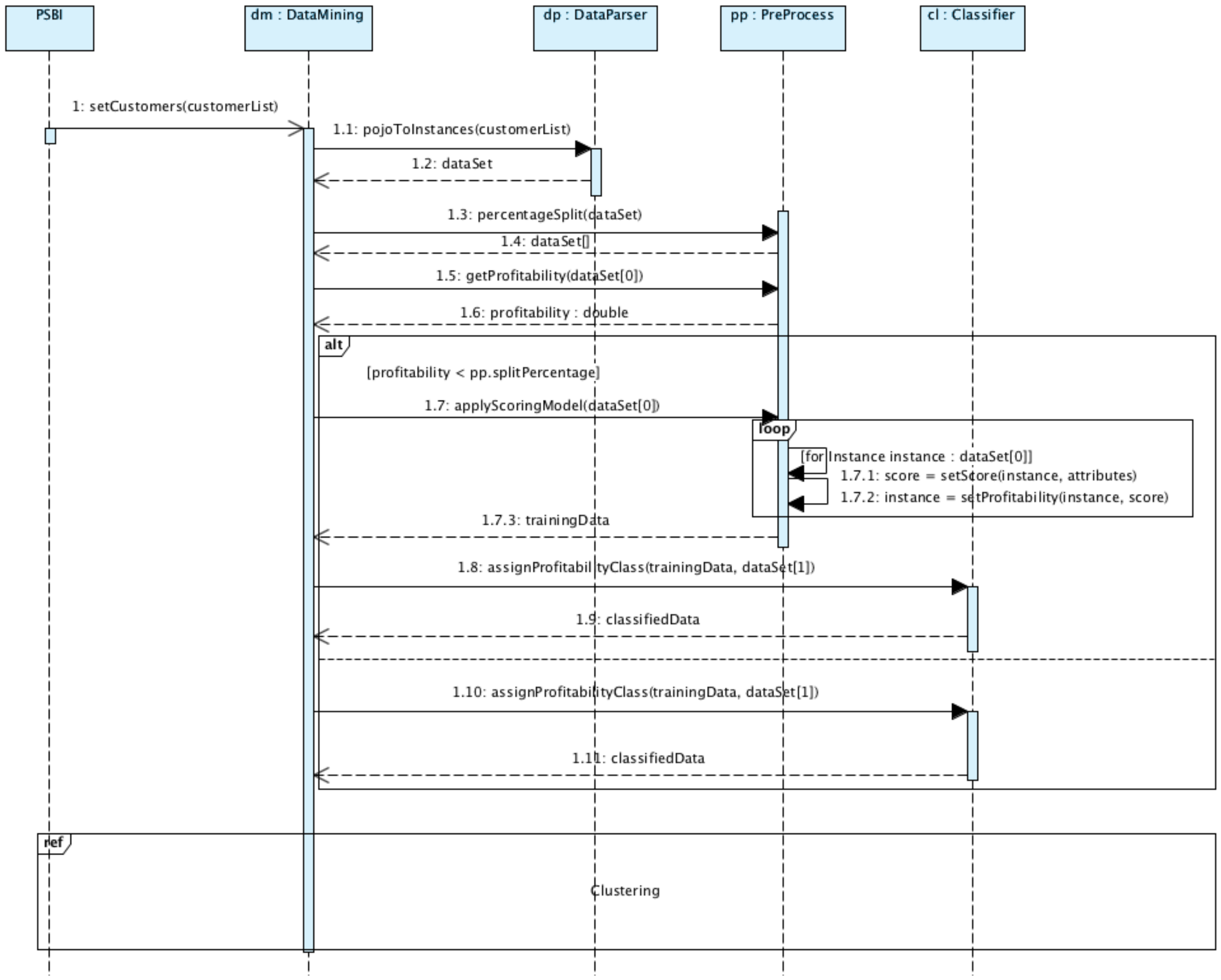

PSBI
dm : DataMining
dp : DataParser
pp : PreProcess
cl : Classifier
1: setCustomers(customerList)
1.1: pojoToInstances(customerList)
1.2: dataSet
1.3: percentageSplit(dataSet)
1.4: dataSet[]
1.5: getProfitability(dataSet[0])
1.6: profitability : double
alt
[profitability < pp.splitPercentage]
1.7: applyScoringModel(dataSet[0])
loop
[for Instance instance : dataSet[0]]
1.7.1: score = setScore(instance, attributes)
1.7.2: instance = setProfitability(instance, score)
1.7.3: trainingData
1.8: assignProfitabilityClass(trainingData, dataSet[1])
1.9: classifiedData
1.10: assignProfitabilityClass(trainingData, dataSet[1])
1.11: classifiedData
ref
Clustering

Fig. 5. UML: Classification

statistics or machine learning. Since our user profile was a business strategist rather than a statistical analyst, we opted out user interaction and decided to tailor a solution to the identified problem. This was the result of gained knowledge of DM, but also a choice based on what we considered vital for the MVP. Such alterations to the scope occasionally appeared in this project due to our initially limited understanding of the field of DM.

B. *Evaluation*

The design review yielded the notion of "Accepted with minor changes". We did the review at a late stage but the changes suggested were insignificant enough to correct without the need of further validation. The most significant suggestion that the review yielded regarded the scoring model, and resulted in the creation of acceptance criteria AC-4 as well as the alternative fragment in Figure 5. The expressed suggestion regarded the inefficiency of calling the scoring module every time, even when the dataset could consist almost entirely of customers with their profitability attribute already set. Thus we created the conditional fragment. Once the changes were made we considered the design accepted.

The reviewer of the review concluded the design to be well defined, with a clear aim and purpose. Simplistic and efficient, and overall correct. In order for a requirement to be considered fulfilled, the acceptance criteria that are related to it must be fulfilled. During the design review we would let the reviewer trace each requirement to design entities to confirms its fulfillment.

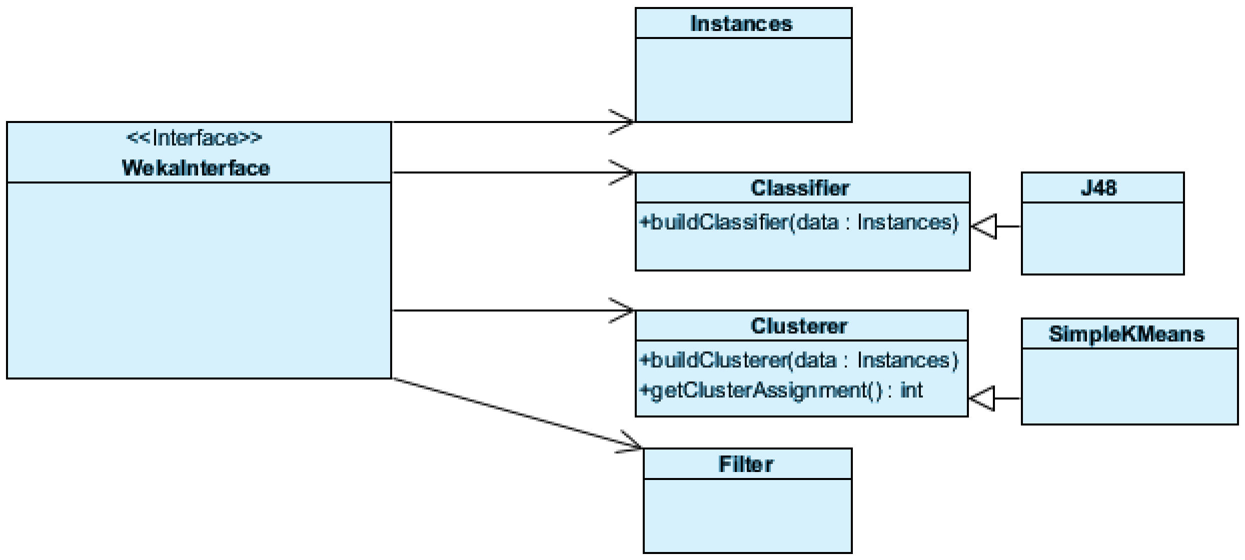
REQ-1

AC-1 Figure 2 depicts the interface offered from PSBI as well as the class that represents each customer.

REQ-2

AC-2 Figure 5, action 1.7 depicts how the scoring model is called using the subset dataSet[0] yielded by action 1.3, percentageSplit.

AC-3 Figure 5, the "alt" fragment is conditional, where the scoring is only performed when the dataset holds a lesser percentage of entities with a profitability



<<Interface>>
WekaInterface
Instances
Classifier
+buildClassifier(data : Instances)
J48
Clusterer
+buildClusterer(data : Instances)
+getClusterAssignment() : int
SimpleKMeans
Filter

Fig. 6. UML: WekaInterface

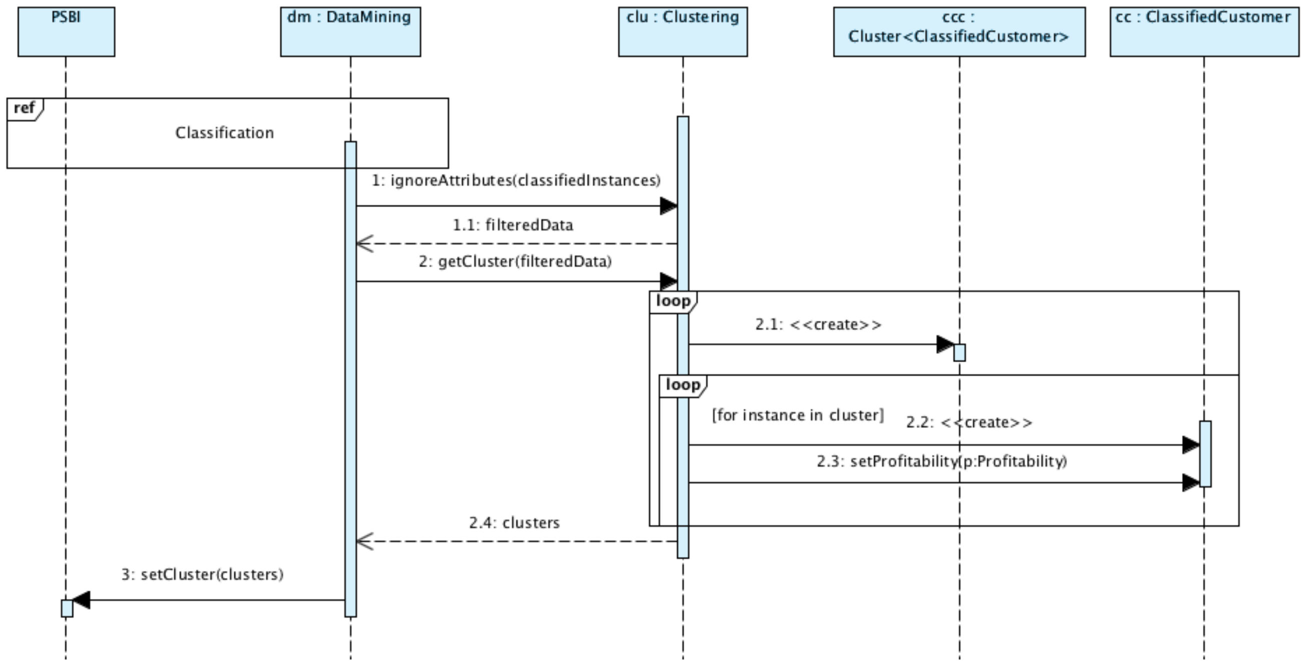


Fig. 7. UML: Clustering

attribute than the constant splitPercentage in PreProcess.

REQ-3

AC-4 Figure 5, action 1.8 / 1.10, method assignProfitability is called with separate datasets for training and test data.

REQ-4

AC-5 Figure 7, action 2.2 and 2.3. For each instance in the dataset, a ClassifiedCustomer is created and the attribute profitability is set. This attribute is modeled in the interface ClusterInterface as design entity *Profitability*.

REQ-5

AC-6 Figure 7, action 1 and 2, 2.1, and 2.4.

AC-7 While what actual attributes used could be considered implementation details, the class Clustering in the class diagram in Figure 4 do contain the method ignoreAttribute which allows filtering of all attributes not relevant for the clustering. This is done prior to execution of getClusters as specified in sequence diagram Clustering (Figure 7), action 1.

REQ-6

AC-8 As seen in Figure 3, the clustered are offered in interface ClusterInterface as Cluster, a generic Collection type.

REQ-7

AC-9 With Weka as the only external library, we are not imposing any licenses on the PSBI except for GNU General Public License [15].

V. Discussion

A. RQ: How can we design a Decision Support System with incorporated data mining?

We presented the design as a Minimum Viable Product (MVP), a bare minimum of what could be integrated in a system of this sort, with the intention of proving a concept. With all the requirements fulfilled, we consider the design to be what we set out to do - a black box component that provide decision support when integrated in PSBI.

When a component is designed to be used with an existing system, one conclusion from this project is that the integration is an essential matters that must be paid attention to. In our case that concern boiled down to two important design aspects; interfaces and external dependencies. In this case they are very much related, as we needed the Weka library there was no question we needed to work around the data casting to meet the requirement on the interface(REQ-6). It might seem myopic to evaluate a design based on such few requirements, but we judge that with our clear definition of the problem, the resulting design may be evaluated on requirements that are equally well defined, and rich in purpose rather than numerous. With that said we consider the design evaluated on fair grounds.

As stated in the Section II, one of the main motivators for this DM approach to the identified problem was Purple Scout's ambition to learn more about DM. Thus, the setting of this project has been largely exploratory, but still we make the argument that the proposed design is a suitable and appropriate solution to the problem at hand, a proof of concept. We motivate this conclusion with the relative simplicity of the design in respect to the problem. The powerful library that is Weka enabled us to perform complex operations with a modest share of time and effort spent on actual development. However, significant time was spent exploring Weka and studying data mining, and even so we are convinced that we have merely scratched the surface of this exciting technology.

B. Impact

The vision of this project was to provide a simple solution for a complex task. As the client still, in 2015, has not yet incorporated an intelligent manner of planning fiber areas, it is possible that providers in other countries have not yet done so either. A simple design that proves effective and that was developed and assessed in less than 3 months could entice colleges to follow. The process of deploying fiber is likely a large and bureaucratic process, but if this design yields satisfying results for a market strategy, then the full process could be sped up and thus made cheaper. This could have a great impact on connectivity, where both provider and customer benefit alike from fiber installed faster and cheaper.

VI. Conclusions

We have developed a design solution to enable decision support by performing data mining operations on customer data in the field of fiber infrastructure. Our aim was to develop a system to help address the problem of identifying profitable areas to deploy fiber, and our design was developed for this purpose. The design was evaluated against a set of well defined criterias and through a review process, and serves as a simplistic proof of concept to answer the identified problem. By profiling customers based on a simple profitability model and classification methods, and by generating clusters of customers with high profitability, our design offers a simple yet suitable approach to decision support in the market specified.

A. Limitations

On a related note, a limitation of this paper is the narrow scope we resorted to as our understanding of data mining grew. Admittedly, the design is evaluated for a very specific business need and while it is of desired quality, it tells us only about designing a specific component for a specific problem, based on the requirements of a specific system. With that we conclude that any conclusions on general DSS with DM design based on the findings in this paper would be vague.

B. Further Research

As we concluded that this design assignment arrived at a narrow scope, we would recommend further research on similar design solutions in different scopes. An example for such research could be the implementation of this design in a fully equipped DSS with a data warehouse and a front end with OLAP functionalities. With similar logic, the DSS could then produce historical data for projects employed in its designated industry, which could prove powerful for evaluation of the projects as well as the theories that founded the design.

VII. Acknowledgements

Special thanks to Jan-Philipp Steghfer for his extensive support in writing this thesis.

References

[1] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. “From data mining to knowledge discovery in databases”. In: *AI magazine* 17.3 (1996), p. 37.

[2] Mehmed Kantardzic. *Data mining: concepts, models, methods, and algorithms*. John Wiley & Sons, 2011.

[3] Ian H Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[4] Jung P Shim et al. “Past, present, and future of decision support technology”. In: *Decision support systems* 33.2 (2002), pp. 111–126.

[5] Kwok-Wing Chau et al. “Application of data warehouse and decision support system in construction management”. In: *Automation in construction* 12.2 (2003), pp. 213–224.

[6] Fan Zhang et al. “Intelligent decision support system based on data mining: Foreign trading case study”. In: *Control and Automation, 2007. ICCA 2007. IEEE International Conference on*. IEEE. 2007, pp. 1487–1491.

[7] G Anthony Gorry and MS Scott Morton. “A framework for management information systems”. In: *Sloan Management Review* 30.3 (1989), pp. 49–61.

[8] Irina Ionita and Liviu Ionita. “A decision support based on data mining in e-banking”. In: *Roedunet International Conference (RoEduNet), 2011 10th*. IEEE. 2011, pp. 1–5.

[9] *Pentaho*. URL: `http://www.pentaho.com/` (visited on 03/19/2015).

[10] Salvatore T March and Veda C Storey. “Design science in the information systems discipline: an introduction to the special issue on design science research”. In: *Management Information Systems Quarterly* 32.4 (2008), p. 6.

[11] R Hevner von Alan et al. “Design science in information systems research”. In: *MIS quarterly* 28.1 (2004), pp. 75–105.

[12] I. Sommerville. *Software Engineering, 9th ed.* Pearson, 20011.

[13] *University of MinnesotaDesign Review: Checklist*. `www.uservices.umn.edu/pmo/assets/docs/04_Design/CHECKLIST_Design_Review.doc`. Accessed: 2015-05-20.

[14] *Weka Sourceforge Weka Documentation*. `http://weka.sourceforge.net/doc.dev/weka/core/Instance.html`. Accessed: 2015-05-20.

[15] *Can I use WEKA in my commercial applications?* URL: `https://weka.wikispaces.com/Can+I+use+WEKA+in+commercial+applications%3F` (visited on 06/01/2015).

Appendix

General Design	*OK*	*Comment*
Does the design support both product and project goals?	☐	
Is the design feasible from a technology standpoint?	☐	
Does the design support proceeding to the next development step?	☐	
Design Considerations	*OK*	*Comment*
Is the design as simple as possible?	☐	
Does the design have conceptual integrity (i.e., does the design tie together)?	☐	
Can the design be implemented within technology and environmental constraints?	☐	
Is the design lean(i.e., are all parts strictly necessary)?	☐	
Does the design allow for scalability?	☐	
Design Convention	*OK*	*Comment*
Are the following attributes well-defined for each design entity?		
Identification Type Purpose Function Interface	☐ ☐ ☐ ☐ ☐	
Design Completeness	*OK*	*Comment*
Are all interfaces described in detail?	☐	
Have implementation details been avoided?	☐	
Are the relationships between entities described?	☐	
Are all relevant architectural views documented?	☐	
Requirements Traceability	*OK*	*Comment*
Does the design address all issues from the requirements?	☐	
Does the design add features or functionality that was not specified by the requirements?	☐	
Are all the assumptions, constraints, design desicions and dependencies documented?	☐	
Have all interfacing systems been identified?	☐	
Compliance	*OK*	*Comment*
Have all legal requirements been assessed and accounted for?	☐	