T.C.

ALTINBAS UNIVERSITY

Graduate School of Science and Engineering

Electrical and Computer Engineering

APPLYING TEXT MINING CLASSIFICATION FOR SOFTWARE REQUIREMENTS PRIORITIZATION

Alialhadi Khaleel Ismael

Master Thesis

Supervisor Asst. Prof. Dr. Sefer KURNAZ

Istanbul, 2019

APPLYING TEXT MINING CLASSIFICATION FOR SOFTWARE REQUIREMENTS PRIORITIZATION

by

Alialhadi Khaleel Ismael

Electrical and Computer Engineering

Submitted to the Graduate School of Science and

Engineering in partial fulfillment of the

requirements for the degree of Master of Science

ALTINBAŞ UNIVERSITY

2019

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science

Asst. Prof. Dr. Sefer KURNAZ
Supervisor

Examining Committee Members (first name belongs to the chairperson of the jury and the second name belongs to supervisor)

Prof. Dr. Mesut RAZBONYALI	Graduation School of Science and Engineering, Maltepe University	
Asst. Prof. Dr. Sefer KURNAZ	School of Engineering and Natural Sciences, Altinbas University	
Assoc. Prof. Dr. Yasa EKŞİOĞLU ÖZOK	School of Engineering and Natural Sciences, Altinbas University	

Prof. Dr. Oğuz BAYAT
Director

I certify that this

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Alialhadi Khaleel Ismael

DEDICATION

First and foremost, I would like to thank Allah Almighty for giving me the knowledge, ability and opportunity to undertake this research study and to persevere and complete it satisfactorily. Heartfelt thanks goes to my father and my mother. Every success is a direct consequence of their influence in my life and their love. At the end I have to mention my family for their support and love.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Supervisor Dr. Sefer KURNAZ for all the knowledge and support he provided during my study for the Master Degree and throughout the work to complete this thesis and I have to mention the kindness and the support to all my friend **MOHAMMED RAOOF** which did not leave me alone the whole time at the courses and while doing this thesis.

ABSTRACT

APPLYING TEXT MINING CLASSIFICATION FOR SOFTWARE REQUIREMENTS PRIORITIZATION

Alialhadi Khaleel Ismael

M.S, Electrical and Computer Engineering, Altınbaş University

Supervisor:Asst Prof.Dr. Sefer Kurnaz

Date: 7/2019

Pages: 66

Software product Priorities are needed in order to discover the most important parts of the product to be developed at first, as the capacity of the development team is constrained by the number of the workers involved, the experience of the individuals, the same as the good coordination of the whole team and the necessity to detect the conflicts among the requirements and planning the sequence of the requirements to be implemented and released. In the this paper , we have review the application of requirements prioritization automation by means of machine learning to the open source applications and software.As the development in the field of ML proceeds, there are many possibilities to utilize the advancements. One of them is the automation of analytic work performed up to now by humans. Requirements prioritization is one of them. It is time consuming and knowledge demanding activity and with the growing number of requirements it can get easily unfeasible for a human being to evaluate every requirement. The first part of this study is the literature review and related works in the field requirements prioritization automation. the second part we have reviewed some text mining background, the third discusses the relevant mechanics and problems of the open source software (OSS) projects.We have analyzed the options for solving the problems of OSS projects requirements management with particular emphasis on the ASF OSS projects, which are interesting with their open development approach. in the fourth part, we propose a system design, using the synthetic methodology of combining different known approaches to solve the particular problems of the OSS projects and the automation of the requirements prioritization process. the fifth part is devoted to the evaluation of the proposed system design with a prototype on apache software

foundation (ASF) hadoop project.The proposed solution and evaluation is valid only in the context of one particular foundation of projects (ASF) and one project (Hadoop). The evaluation outcome cannot be generalized, since the fine tuning of the algorithms would require enormous effort. Using another project would mean the refitting of the whole proposed solution.

Keywords: Requirements Prioritization OpenSource Software Apache Software Foundation Machine Learning Text Mining, Support Vector Machine,Support Vector Regression.

ÖZET

YAZILIM GEREKLİLİKLERİ ÖNCELİKLERİ İÇİN METİN MADENCİLİK SINIFLANDIRMA UYGULAMASI

Alialhadi Khaleel Ismael

Yüksek Lisans, Elektrik ve Bilgisayar Mühendisliği, Altınbaş Üniversitesi

Tez Danışman: Yrd.Doç. Dr. Sefer Kurnaz

Tarih: 7/2019

Sayfalar: 66

Yazılım ürünü. geliştirme ekibinin kapasitesi, ilgili çalışanların sayısıyla, kişilerin deneyimlerinin, iyi bir koordinasyonla aynı şekilde kısıtlanmasından dolayı, ilk olarak geliştirilecek ürünün en önemli kısımlarını keşfetmek için önceliklere ihtiyaç vardır. tüm ekip ve gereksinimler arasındaki uyuşmazlıkların tespit edilmesi ve uygulanacak ve serbest bırakılacak gereksinimlerin sırasını planlama zorunluluğu bu yazıda, açık öncelikli uygulamalara ve yazılımlara makine öğrenmesi yoluyla gereksinim önceliklendirme otomasyonunun uygulanmasını gözden geçirdik. ml alanındaki gelişmeler ilerledikçe, bu gelişmelerden yararlanmanın birçok yolu var. bunlardan biri, şimdiye kadar insanlar tarafından gerçekleştirilen analitik işlerin otomasyonu. gereksinimler önceliklendirme bunlardan biridir. zaman alıcı ve bilgidir zorlu faaliyetler ve artan sayıda gereksinim ile bir insanın her gereksinimi değerlendirmesinde kolayca olanaksız hale gelebilir. bu çalışmanın ilk kısmı, alan gereksinimleri önceliklendirme otomasyonu ile ilgili literatür taraması ve ilgili çalışmalardır. ikinci bölümde bazı metin madenciliği geçmişini inceledik, üçüncüsü açık kaynaklı yazılım (oss) projelerinin ilgili mekaniklerini ve problemlerini tartışıyor. açık gelişim yaklaşımlarıyla ilgi çekici projeler. dördüncü bölümde, oss projelerinin belirli sorunlarını çözmek için bilinen farklı yaklaşımları bir araya getirmenin sentetik metodolojisini ve gereksinim önceliklendirme sürecinin otomasyonunu kullanan bir sistem tasarımı öneriyoruz. beşinci kısım, önerilen sistem tasarımının apache yazılım temeli (asf) hadoop projesi üzerinde bir prototip ile değerlendirilmesine ayrılmıştır. önerilen çözüm ve değerlendirme, yalnızca belirli bir proje temeli (asf) ve bir proje kapsamında geçerlidir. hadoop).

Anahtar Kelimeler: Gereksinimler Önceliklendirme, Açık Kaynak Kodlu Yazılım, Apache Software Foundation Makine Öğrenmesi, Metin Madenciliği, Destek Vektör Makinesi, Destek Vektör Regresyon.

TABLE OF CONTENTS

Pages

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

LIST OF ABBREVIATIONS

OOS	:	Open Source Software
ASF	:	Apache Software Foundation
ML	:	Machine Learning
CBR	:	Case-Based Ranking
AHP	:	Analytical Hierarchy Process
PH	:	Priority Handler
SVM	:	Support Vector Machine
SVR	:	Support Vector Regression
VSM	:	Vector Space Model
TF-IDF	:	Term Frequency – Inverse Document Frequency
RE	:	Requirements Engineering
IM	:	Instant Messaging
IRC	:	Internet Relay Chat
SOA	:	Service Oriented Architecture
DS	:	Data Source
LOC	:	Lines Of Code
ORM	:	Object Relational Mapping

1. INTRODUCTION

1.1 CONTEXT

Requirements prioritization is very important part of the process of developing a new software product. Priorities are needed in order to discover the most important parts of the product to be developed at first, as the capacity of the development team is constrained by the number of the workers involved, the experience of the individuals, the same as the good coordination of the whole team and the neces- sity to detect the conflicts among the requirements and planning the sequence of the requirements to be implemented and released. [1]

The priority itself is a metric attribute of a requirement, that may serve various purposes depending on the needs and should lead to the process of requirements prioritization. The Helsinki University of Technology's Qure (Quality through Requirements) case project,wh- ich lasted for three years, learned that the terms "prioritization pro- cess" and "priority" are not uniformly defined and they have many meanings in the case companies, which leads to misunderstanding and confusion of the team members. The prioritization sometimes means the strategic process of the setting of the priority in the long term and sometimes the operative selection of the most important requirements to be implemented at present for the next release. Or sometimes it means the process of identifying the requirements to be implemented first in the new project. The priority is also ambiguous term as it may signify the importance of the requirement to the cus- tomer while another time it denotes the deadline when it should be finished. Also the priority scales doesn't have set meaning among the team members and it takes a long discussion about them. [1]

In the case companies, there was no common process of require- ments prioritization. The process relied mostly on the basis of per- sonal competencies and experience. There were no explicit methods in use. The personnel in charge made a rough guess with no system- atic approach. The contracts with the customers and informal dis- cussions had the major influence on the priorities and the companies often got into

situations of just trying to avoid the fines for violating the contracted deadlines. [1] In practice, there was no time to analyze the raw requirements data they gathered from the customers and discover the relevant in- formation needed for sound priority decisions. The priorities assigned to the requirements were based on the cost-value analysis, which were very informal and the product managers weren't able to truly com- bine information from different sources. [1]

1.2 PROBLEM STATEMENT

As we can see, the problem of requirements prioritization is a big issue in the world of software engineering. This topic is widely dis- cussed and the volume of research being conducted is increasing with each year passing. The problem up to now lies in the fact, that existing techniques for requirements prioritization are can only handle sam- ple toy projects or on the other hand are too complex to be success- fully implemented in the real world scenario, yet with unconvincing results. [2] The issue is related right to the fact, that OSS projects have distributed nature.

There are two main problems related to the automation of require- ments prioritization in the context of OSS projects:

1.The majority of requirements prioritization research focuses on solving standard proprietary projects problems. How ever OSS community projects are being managed in a different way and therefore require different approach to solve the requirements prioritization automation problem. For more information, see the section 4 *Open source software* projects.

2. According to A. Guzzi et al. [3], the OSS projects have a big problem with communication, which is scattered among infor- mation repositories and the project developers have problems with maintaining and refreshing all the information about the project.In practice, there was no time to analyze the raw requirements data they gathered from the customers and discover the relevant in- formation needed for sound priority decisions. The priorities assigned to the requirements were based on the cost-value analysis, which were very informal and the product managers weren't able to truly com- bine information from different sources. [1]

1.3 IMPORTANCE OF THE PROBLEM

The related work mainly focuses on requirements prioritization in closed source projects. While the open source projects me- chanics work differently.

1. The problem of automatic information linking between different sources, is also crucial for the automation of the require- ments prioritization process, because we can automate the pri- oritization process, only if we have all the needed information at one place. Also the project community members would be freed from doing this task manually, so the overall productivity of the project could rise. [3]

Requirements prioritization is an important and complex phase in the software engineering process that leads to the decision about the most important requirements to be implemented first. An aid to the automated decision about the priority of requirements has been pro- posed in recent research[5], but its application still needs validation, especially in OSS projects that collect large number of requests from the community.The first part of the problem lies in the fact, that the experiment[5] ihad proprietary data set and on top of that, some of the exact set- tings of the used algorithms were not published in the paper. Thus it is not easily possible to replicate this experiment with the same set- tings (like the same data set) to for example benchmark the original algorithms performances against the new or different ones. The sec- ond part is related to the applicability of such *Machine learning* (ML) approach to large OSS projects.

1.4. OBJECTIVES

The first objective is to review current state of the art in the classifica- tion and prediction of priorities in software engineering to set up the base–ground for our next work.

The next objective is to analyze the way, how OSS projects are be- ing managed (especially in the terms of priority management of the requirements)

The primary objective is to study and implement the machine learn- ing approach in the context of large OSS projects and to validate if this approach could be applicable to such projects.

1.5 CONTRIBUTION

In this thesis, we address the problem of using machine learning ap- proaches for requirements prioritization in the context of OSS. We analyzed the current state of the art in this research field and stated, that there is no solid research in the contextof the OSS projects. Given the results of the previous research work, we used a synthe- sis way to combine different approaches and algorithms to solve the problem of requirements prioritization in order to design a system for our scenario. The next step was the evaluation of such design in the context of real open source project and adapting the original design to real world scenarios. So the first contribution is adaptation of the original approaches posted by former researchers to the context od OSS projects. The next contribution is the evaluation of the projects dynamics in time by means of communication. The third contribu- tion is building up the prototype for mining data from JIRA and mail lists for further processing

2. AUTOMATION OF REQUIREMENTS PRIORITIZATION

2.1 LITERATURE REVIEW

As the development in the field of ML proceeds, there are many possibilities to utilize the advancements. One of them is the automation of analytic work performed up to now by humans. Requirements prioritization is one of them. It is time consuming and knowledge demanding activity and with the growing number of requirements it can get easily unfeasible for a human being to evaluate every requirement. The ML approach is trying to solve this problem by exploiting and learning domain knowledge and utilizing it to automatically preprocess the requirements and assign some basic priority. So the human decision-maker who is responsible for the assignment of the priority can focus only on the important requirements.

2013 Systematic mapping study performed by the authors M.Per-gher and B. Rossi [4] reviewed the requirements prioritization studies with the focus on empirical studies. The results show, that the re- search in the field of requirements engineering is increasing in the recent years. The majority of the analyzed studies tried to find the most accurate technique for prioritizing (compared to an theoretical optimal ranking), with lesser regards to the other practical questions like scalability, fault tolerance and practical usability. The evaluators were frequently using sample data for evaluation. The paper points out, that the research concerning the possible integration of proposed prioritization techniques into existing *Requirements Engineering* (RE) techniques and process is an unexplored area.The experiment conducted by A. Perini, A. Susi, and P.Avesani [5] was one of the first attempts to study the automated requirements prioritization process with the application of ML on the real world data–set. The paper discusses the two different ways to deal with this problem – the *ex-ante* and *ex-post* approach. The ex-ante approach tries to formulate the target criterion for ranking requirements in advance and then assigns the rankings according to the ranking according to the requirements attributes. The problem of this approach is the independence of the target ranking criterion and the examined set of requirements. On

the other hand, the ex-post approach tries to utilize the known solutions to similar problems. The paper focuses on the ex-post approach, namely Case-Based Reasoning. involve the stakeholders, which have to set relative pairwise priority among pairs of requirements. Therefore the final ranks are not computed from the (absolute) values of the requirement attributes, but by the assigned (relative) rankings. The relative rankings are considered to have lesser input noise than the absolute values. And the stake- holders might take their decision based on some implicit information, which may not be encoded directly in the requirement's attributes. The paper introduces *Analytical Hierarchy Process* (AHP), which is used as a reference method. AHP uses pairwise comparison between all requirements pairs. This approach is in theory the most accurate (every pair is being evaluated), but the real usability of this approach is get- ting worse and worse with the increasing number of requirements, as the number of pairs rises quadratically. So the paper proposes a new better approach, called *Case-Based Ranking* (CBRank). This approach allows for *domain adaptability* – combining sets of priorities elicited by human input and the automatically computed priorities using the ML approach (which utilizes the partial pairwise priorities) to minimize the number of the needed human input comparisons. The CBRank is directly applicable to different domains and that the estimated priority accuracy increases with the amount of encoded information. **2014** One year later, the researchers P. Achimugu, A. Selamat,R. Ib-rahim and Mohd Naz'ri MAHRIN conducted a systematic review of the state of the art of the requirements prioritization research [6]. The method consisted of systematic search, which led to the selection of 73 relevant studies, including the work of A. Perini's team [5], and their analysis. Concluding, that the prioritization is significantly discussed problem in the requirements engineering domain. Stating that the existing prioritization techniques suffer several limitations, including scalability concerns, requirements' evolution leading to rank up- dates, stakeholders coordination and all in all, the existing techniques are complex and the real usage is yet to be reported.

Paper based on the results of [6] also conducted by the same team of P. Achimugu, A. Selamat, and R. Ibrahim [7] focuses on the possibilities to requirements prioritization automation. Discusses the related work and possible approaches on this topic and the

problems of the approaches. There are mentioned two main categories of prioritization techniques: The first allows the stakeholders to assign weights (priority) to the requirements (e.g. AHP and CBRank), while the other makes use of the communication between the stakeholders to negotiate an agreement on the priorities (e.g. MoSCoW and planning game). While the AHP is recognized as the most popular approach used in other related works and considered to be the best in the terms of reliability, the limitations of scalability and the inability to rank newly added requirements are not useful for a real world The main focus of this paper is based on the statement, that while there are many methods for requirements prioritization, the support tool for real usage is non-existent, so the paper proposes a web-based multi criteria decision making tool to help the stakeholders with the prioritization process. The proposed tool has generic architecture which is supposed to be used for an ML enhanced requirements prioritization. The system is based on ex-post ranking of the usage requirements on a importance scale of 1(low) - 5(high), assigned by the stake holder while the multiple criteria decision making system computes the relative weights. When the requirement changes, the priority is re- computed, based on the linear combination of the requirement's at- tributes (compared with the other known requirements). Still, the paper lacked the implementation details of the system.

2015 Another year later M.I. Babar et al. [8] published a relevant paper, which also highlighted the fact, that existing requirements prioritization techniques are not suitable for large number of requirements and handle only toy projects or projects with small number of requirements. The paper identifies main problems of existing techniques as following:not scalable,not sufficient level of automation, not intelligent enough, time consuming, complex = difficult to imple- ment and still leading to faulty results.

The paper aims to solve the first problem – the scalability concern, proposing a new approach called *Priority Handler* (PHandler), consisting of a combination of three techniques. At first value-based technique using expert knowledge, a back-propagation neural net- work to predict a priority and a traditional AHP, which is applied to prioritized groups of requirements to be scalable. this approach is

completely different to the approach of [7].

2016 Two years later after the original P. Achimugu et al. paper [7], this team further developed and fully implemented the originally proposed system, which is called *Requirements Prioritizer* (ReproTizer).

[9] The paper notes, that the PHandler approach has the same prob- lem as CBRank as it is unable to update rankings with newly added or deleted requirements.

requirements on a importance scale of 1(low) - 5(high), assigned by the stakehold- ers, while the multiple criteria decision making system computes the relative weights. When the requirement changes, the priority is re- computed, based on the linear combination of the requirement's at- tributes (compared with the other known requirements). Still, the pa- per lacked the implementation details of the system.

2015 Another year later M.I. Babar et al. [8] published a relevant paper, which also highlighted the fact, that existing requirements pri- oritization techniques are not suitable for large number of require- ments and handle only toy projects or projects with small number of requirements. The paper identifies main problems of existing tech- niques as following: not scalable, not sufficient level of automation, not intelligent enough, time consuming, complex = difficult to imple- ment and still leading to faulty results.

The paper aims to solve the first problem – the scalability con- cern, proposing a new approach called *Priority Handler* (PHandler), consisting of a combination of three techniques. At first value-based technique using expert knowledge, a back-propagation neural net- work to predict a priority and a traditional AHP, which is applied to prioritized groups of requirements to be scalable. this approach is completely different to the approach of [7].

`

2016 Two years later after the original P. Achimugu et al. paper [7], this team further developed and fully implemented the originally proposed system, which is called *Requirements Prioritizer* (ReproTizer).

[9] The paper notes, that the PHandler approach has the same prob- lem as CBRank as it is unable to update rankings with newly added or deleted requirements.

3. TEXT MINING

Text mining is the process of information retrieval from natural lan- guage text documents. It consists of several approaches, which will be further introduced in this section.

3.1 TEXT DOCUMENT REPRESENTATION - FEATURE EXTRACTION

Text documents cannot be directly interpreted by the ML algorithm, so a proper representation is necessary. Usual approach is a *Vector Space Model* (VSM).The basic model of vectorization term frequency (occurence count- ing) is *Bag-of-words*, where a document d is represented by a vector of weights $d = < w_1, w_2, \ldots, w_V >$, where V is a set of terms that occur at least once in the training document set D. [23]The updated model to the The Bag-of-words is the *Bag-of-ngrams*. Which incorporates n-grams into the VSM. *N-gram* is a sequence of n words contained in the document. Bag of 1-grams (unigrams) is directly equal to the bag of words. The 2-grams (bigrams) consists of two following words.[23] $f(t, d) = f requency(t, d)$

3.1.1 TF-IDF

The *Term Frequency – Inverse Document Frequency* (TF-IDF) is the im- provement to the basic TF - term frequency - word count vectoriza- tion (Bag-of-words). As the number of documents grow bigger, the occurence of some words will be very large. In basic count vectoriza- tion that means, they have more influence. But in reality these terms do not bring any new information to the model as there is no variance between documents. So we want to lower the priority of these words in contrast to the priority of rarely occuring words.[23]contrast to the priority of.

3.2 CLASSIFICATION

Text classification is being used for categorizing text documents in a predefined set of classes, based on the document content. Text clas- sification dates back to 1960's, but used expert knowledge based ap- proaches, which relies on manually defined rules of classification. The ML based approaches didn't appear until the late 1980', but then become pervasive. These approaches incorporate the ML techniques to automatically build text classifiers based on learning from manu- ally classified documents. [22]

3.3 CLASSIFICATION FUNCTION

Formally, the text classification function f can be defined as followed: Let's have an instance space X, where each instance is a text document d and a fixed set of classes $C = \{C_1, C_2, \ldots C_{|C|}\}$, where C is the number of classes. Given the training set D of training documents $< d, C_i > \; X \; C$, the goal is to use the training set of documents D to construct a classification function f with the training documents to classify any document: $f(d) : X \rightarrow C; \forall d \in X$ [23]

3.4 REGRESSION

The opposite approach to classification is regression. While classifi- cation tries to assign known class to a model, the regression function tries to predict some continuous or ordered variable. So the problem is to find the relation between variables of the model.[11] The regres- sion problem can be defined as $y \approx \beta X$. Where we are searching for the β parameter, which signifies the relation between y and X.

3.5 PERFORMANCE MEASURE

Recall, *precision* and F_1-measure is the most popular measure for eval- uating the performance of a text classifier.[22] Given a classifier, whose input is a document and output a ranked list of categories assigned to such document.

defined decision threshold:

$$Recall = \frac{categories\ Found\ AND\ Correct}{total\ categories\ Correct} \tag{3.1}$$

$$Precision = \frac{categories\ Found\ AND\ Correct}{total\ categories\ Found} \tag{3.2}$$

where „categories Found" means categories above the threshold.[24] The F_1-measure is traditionally defined as a harmonic mean of *precision* and *recall*: [24]

$$F_1 = 2 \frac{precision \cdot recall}{precision + recall} \tag{3.3}$$

3.5.1 Kendall Tau Significance Test

This measure shows the pairwise correspondence of two rankings. Kendall tau is defined as[32]:

$$\tau = \frac{c-d}{\sqrt{(c+d+t)\,(c+d+u)}} \tag{3.4}$$

where

c = concordant pairs, d = discordant pairs, t = ties in first rank- ing and u = ties in second ranking.[32]

$f(\tau) \in <-1; 1>$.

The value 0 means no correlation. 1 means perfect match and -1 means reverse ordering.[32]

3.6 SUPERVISED LEARNING AND UNSUPERVISED LEARNING

The majority of ML methods incorporate the supervised learning. The supervised learning uses the training data to supervise the learn- ing process of the ML method to predict the output. While the unsu- pervised learning method doesn't incorporate the training data in the process. The example of supervised learning algorithm might be su- pervised classifying with labeled training dataset of some data, while the unsupervised algorithm might use clustering instead of classifi- cation, so if the dataset has good data, the clustering method might be able to separate the different sets of the data, but would not tell us the semantics (a.k.a. the classes in the supervised method).[11]

3.7 SVM

The recent research in the field of machine learning led to a new gen- eration of algorithms. One of them is SVM, which was successfully used to information retrieval from text. The SVMs can be used both for classification and regression, but are rather used for classification. The SVMs are highly effective in hish dimension vector spaces. [23] The SVM looks at the data as points in n-dimensional space. The n is the number of features of the item. If used for classification, the SVM tries to find a decission surface (hyperplane) which is maximally far away from any data point. [23] The SVM method is formally de- scribed as:[30] Given the training set of n-dimensional vectors $x_i \in R^n$, $i = 1, \ldots, l$, in two classes, and a vector $y \in R^l$ such that $y_i \in \{1, -1\}$, C-SVM solves the following pri- mal problem:

$$\min_{w,b,\xi} \quad \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \tag{3.5}$$

$$y_i(w^T \varphi(x_i) + b) \geq 1 - \xi_i,$$

$$\xi_i \geq 0, i = 1, \ldots, l.$$

This primal problem can be transformet into the dual prob- lem defined as:

$$\min_{a} \quad \frac{1}{2} a^T Q a - e^T a$$

$$0 \leq a_i \leq C, \qquad i = 1, \ldots, l, \tag{3.6}$$

$$y^T a = 0,$$

where e is the vector of all ones, C is the upper bound, Q is an l by l positive semidefinite matrix, $Q_{ij} \equiv y_i y_j K(x_i, x_j)$, and $K(x_i, x_j) \equiv \varphi(x_i)^T \varphi(x_j)$ is the kernel.

3.7.1 SVM regression

A SVM method can be used not only for classification, but it can be extended to also solve regression problems. The SVR performs linear
regression in n-dimensional space using ϵ loss function. The method is called SVR.[29] The SVR method is formally described as: [30]

Given the training set of n-dimensional vectors $x_i \in \mathbf{R}^n$, (x_l, z_l) , such that $x_i \in R^n$ is an input and $z_i \in R^1$ is a target output, the support vector is defined as minimal- ization problem:

$$\min_{w,b,\xi,\xi^*} \quad \frac{2}{} + \sum_{i=1} \xi_i + \sum_{i=1} \xi_i \tag{3.7}$$

$$z_i - w^T \varphi(x_i) - b \leq \epsilon + \xi_i, \; w^T$$
$$\varphi(x_i) + b - z_i \leq \epsilon + \xi_i^*, \xi_i, \xi_i^*$$
$$\geq 0, i = 1, \ldots, l.$$

The primary minimalization problem can be transformed to this dual

3.8 HYPERPARAMETER TUNING

The hyperparameters are the parameters, which are not directly learnt from by the predictor methods by the training data. For getting best prediction accuracy, it is recommended to do a hyperparameter tun- ing. This can be done by searching the parameter space for optimal values. The two basic methods are random search and grid search. The random search samples the space given by some random dis- tribution, while the grid does an exhaustive search of the parameter space based on the selected grid intervals. This approach can have a significant influence on the predicted values, but is computional challenging.[31]

3.9 USING TEXT MINING IN THE CONTEXT OF REQUIREMENTS

There are different studies, that look into requirements retrieval from text. We will look at some of them in order to get basic background in this area.

The paper of M. Xiao et al. [10] concerns the problem of automatic mining useful information from the Q&A sites. The paper focuses on acquiring feature requests from the users' posts. The approach consists of a SVM using TF-IDF approach to compare different text documents (posts), combined with a dictionary of requirements key- words. To evaluate the SVM approach, the researchers implemented also prototype of another technique, consisting of a set of linguistic rules. The evaluation showed, that the linguistic rules approach led to worse results (F-measure 57.8%) compared to the results of using pure SVM only, without the dictionary (F-measure 71.77%). The con- cept of the linguistic rules is easy to understand, but the key prob- lem lies in the fact, that it is not easy to create rules which would fit every way of possible expression of the users' ideas. Also the re- searchers had to build the rules manually, which was exhausting time consuming. On the other hand, the SVM, which led to better results, can be taught by any kind of training data set and is also able to au- tomatically create an appropriate classifier. The SVM enhanced with the keyword dictionary did not lead to significantly better results (F- measure 74.22%), but the researchers attribute this result to the small size of the dictionary

4. OPEN SOURCE SOFTWARE PROJECTS

In this chapter, we will discuss some OSS specifics to show, that the context of proprietary software development is way different than the context of large OSS projects.This section summarizes the pecularities of the OSS projects which may affect (directly or indirectly) the requirements prioritization.

4.1 GOVERNANCE MODELS

There are many ways, how to manage OSS projects. They range from the centralized models when the responsibility of the control lies in the hands of a single individual (benevolent dictator) to distributed models, in which the wider counsel makes the decisions(meritocracy). The governance model describes the inner processes of control and also the contribution model of the project. [14]

4.1.1 Contribution Models

The contribution model illustrates wether the project foster the con- tributions from broader community (bazaar model) or utilizes small core of stable contributors (cathedral model). [14]We can find existing projects at any point between the centralized and distributed governance models as the between the bazaar and cathedral models, while the governance model doesn't imply the contribution model and the projects can move along and shift the models as they mature. [14]

4.2 OPEN DEVELOPMENT

In some OSS projects we can note some differences between the terms "Open source" and "Open development". While the source code of the software is publicly available and released under some free soft- ware license. The

problematic part is, that the steps taken and de- cisions made to develop this source code are not public, so anybody who wants to develop the software further doesn't know the story be- hind it. So with the Open development method, the community not only releases the source code, but also the documentation – historical context about all the decisions that had been [20]

who wants to develop the software further doesn't know the story be- hind it. So with the Open development method, the community not only releases the source code, but also the documentation – historical context about all the decisions that had been taken. [20]

4.3 COMMUNICATION IN THE OSS PROJECTS

4.3.1 Mailing Lists

The earlier works reported, that the mailing lists are the core part of the communication in the OSS projects. For example Mockus et al.[15] claimed that the developers communicate only through mail- ing lists, recent research discovers a big change in the project com- munication. A. Guzzi et al. [3] analyzed the developers' mailing lists threads of the OSS project Lucene (which belongs to the ASF just as the Apache Server). The study was analyzing mainly the developer's mailing lists, but found out, that the developers are also monitoring the user's mailing lists in order to better understand the real usage of the developed application. Historically, the mailing lists were consid- ered as the information hub of the OSS projects. This consideration proves false for the modern OSS projects like Lucene, since its make of extensive use of the issue repository (JIRA), where a significant amount of communication is taking place (and still increasing). The developers also use an *Instant messaging* (IM) platform (*Internet Re- lay Chat* (IRC)) to discuss details of the project and implementation. There is also evidence

of numerous developer's personal meetings. The policy of most OSS projects is to archive official messages along with important discussions from different sources to the mailing list, yet there is often clear situation, when this policy gets broken and the information flow between the sources is disconnected. Which leads to coordination issues and information loss or duplicity. Moreover, the paper reveals, that the communication channels work in parallel and disconnected from each other and the project developers have prob- lems with maintaining awareness about each other's work. The paper calls for a tool for automatic linking of relevant information among different data sources, which would help the researchers in obtaining better picture of the development process as like as take some of the strain off the developers.

4.3.2 Community

Crowston and I. Shamshurin [25] examined the relationship be- tween community members and the project success. The study used data from 74 projects of the ASF Incubator. The core members were identified by the official list of project developers, the other were iden- tified as peripheral. The results suggest, that the successful projects (which managed to build a community and graduate from incuba- tion) have more members and a matching amount of communication. The core members contribute more code, but the number of messages is evenly split between the core and peripheral members, suggesting that both roles play an important part in the successfull OSS project.

Hannemann et al. [26] discussed, that while there are plenty of studies concerning OSS project visualization tools, most of them are related either only to the source code or standalone developers. This paper investigates the question of visualizing the evolution of the whole community. The online survey among OSS communities showed, that there is a big interest in such solution. The survey results show, that 75% of the participants use web-based issue-management tool like

GitHub and 63% of the OSS developers were interested in the social community related statistics. The majority of the participants also showed a strong interest in a text mining analysis of the commu- nication for purposes like:"*determine the needs of the users in addition to voting and tagging in bugtrackers*", "*creating FAQs for new contributors*" and "*finding out in which direction the community wants to evolve*". The survey also discovered, which functionalities the users missed in the existing OSS visualization statistics: which problems are the most dis- cussed, non-code contributions (reported by, tested by) statistics and overall more information about project activity. Based on the survey, the researchers developed a prototype Web-based dashboard filled with mailing lists communication data of three OSS Bioinformatics projects. This prototype was sent to members of three communities for evaluation. The most positive feedback received the social net-work graph of the community, which is really interesting for the de- velopers: "*The social aspects of OSS projects are no less intriguing than the technological ones!*" and "*[...] there is a lot to learn from this on how OSS projects get off the ground, what makes a successful project, etc*". The dis- covered weakness is that the data source is limited to the mail lists

as the details are often discussed for example on the issue repos- itory, so adding such data source to the text mining and compar- ing the results with the mail lists could be interesting. K. Crowston and I. Shamshurin [25] examined the relationship between commu- nity members and the project success. The study used data from 74 projects of the ASF Incubator. The core members were identified by the official list of project developers, the other were identified as peripheral. The results suggest, that the successful projects (which man- aged to build a community and graduate from incubation) have more members and a matching amount of communication. The core mem- bers contribute more code, but the number of messages is evenly split between the core and peripheral members, suggesting that both roles play an important part in the successfull OSS project.

A. Hannemann et al. [26] discussed, that while there are plenty of studies concerning OSS project visualization tools, most of them are related either only to the source code or standalone developers. This paper investigates the question of visualizing the evolution of the whole community. The online survey among OSS communities showed, that there is a big interest in such solution. The survey results show, that 75% of the participants use web-based issue-management tool like GitHub and 63% of the OSS developers were interested in the social community related statistics. The majority of the participants also showed a strong interest in a text mining analysis of the commu- nication for purposes like:"*determine the needs of the users in addition to voting and tagging in bugtrackers*", "*creating FAQs for new contributors*" and "*finding out in which direction the community wants to evolve*". The survey also discovered, which functionalities the users missed in the existing OSS visualization statistics: which problems are the most dis- cussed, non-code contributions (reported by, tested by) statistics and overall more information about project activity. Based on the survey, the researchers developed a prototype Web-based dashboard filled with mailing lists communication data of three OSS Bioinformatics projects. This prototype was sent to members of three

communities for evaluation. The most positive feedback received the social network graph of the community, which is really interesting for the de- velopers: "*The social aspects of OSS projects are no less intriguing than the technological ones!*" and "*[...] there is a lot to learn from this on how OSS projects get off the ground, what makes a successful project, etc*". The dis-covered weakness is that the data source is limited to the mail lists as the details are often discussed for example on the issue repository, so adding such data source to the text mining and comparing the results with the mail
lists.could.be.interesting.

4.4 REQUIREMENTS PRIORITIZATION IN THE CONTEXT OF OSS

The distributed nature of OSS is also shaping the project manage- ment style of such projects. There is usually a large number of dif- ferent requirements from different sources. The community dynam- ics as shown in the previous section also plays its role. In the whole, there is not much direct information about OSS projects in regard to the requirements prioritization.

Laurent and Cleland-Huang[13] explored and evaluated the on- line requirement prioritization process. It is rather old paper, but the general idea is still the same. The thing what changed from the time of publishing is by our observation the technology - in 2009, there were online forums for discussion, while today most of the projects facil- itate some online requirement management system (namely JIRA). The paper distinguishes

5. SOLUTION PROPOSAL

This chapter is devoted to the high level solution proposal. We will analyze the options for solving the problems of OSS projects requirements management in this chapter with particular emphasis on the ASF OSS projects, which are interesting with their open development approach. The previous chapters outlined the current development in the fields of requirements prioritization automation, text mining and the review of the OSS projects working principles. This chapter is divided into two sections. The first section is about the problem of information linking from different sources. The second section analyses the possible ways of prioritizing requirements in the OSS projects.

5.1 METHODOLOGY

We apply the synthesis methodology of the previously introduced concepts to propose a system, which would solve the discovered problems.
The identified problems in the context of OSS are following:
1. large amount of requirements in the repositories
2. managing the priorities retrieval from different sources (voting, reviews, users etc.)
3. manual requirement repository management (some projects have outdated requirements repository)
4. manual prioritization process (time consuming and the process gets harder with the expanding number of requirements in the repository)
5. linking data from different data sources and communication channels (e.g. design discussion in IRC, voting in emails, requirement in issue tracking system)
6. managing the ever changing community In the next sections, we will propose a system concerning these problems.

5.2 DATA SOURCES

The first part we need to discuss is information. We can gain infor- mation from different sources. We identified the data sources of the OSS projects of two basic types:

Main data sources of the OSS project are those, which are directly managed by the community. As noted by [3], 88 projects of the ASF use the same standard communication channels: Mailing lists, JIRA Issue repository and IRC channels. We consider also the project web- site and source code repositories as the basic resources. For more info see table 5.1

Additional data sources are all, which do not completely belong to the project itself. Here can be custom search results from the web, QA sites issues or user reviews. For more info see table 5.2 on the next page.

Table 5.1: Main Data Sources

Name	Description
Mailing lists	officially the main mean of communication among the ASF projects. [27], needs text mining for information retrieval
Issue repository	contains structured data aboutissues
Project website	contains official list of core members for rough community structure estimation [26]
IM	Mainly IRC channels are being used in the ASF [27]. Serve as a synchronous communica- tion channel for discussing implementation de- tails and decisions [3].
Source Code	Another source for community social graph es- timation.

5.3 PROCESS VIEW

To address the identified problems we identify following three basic areas we have to handle: data linking, requirement management and community management. Each area consists of some processes.

Table 5.2: Other Data Sources

Name	Description
Competitors	The projects often reflect the competitors' new functions in the next development plan.
Web search	Custom web search results related to some issue
Reviews	We can gain new ideas from user reviews
Q/A Sites	Search information related to the project

5.3.1 Requirement Management

Requirement management is one of the basic components in the sys- tem. It serves the purpose of automation of the part of the (currently) human tasks. Amogst the processes of reqirement management be- longs the requirement prioritization for predicting the priority of the requirements the second is requirement repository management, wh- ich should concern the good condition of the repository

5.3.2 Data Linking

The data linking component is essential for bringing all the related data together. The basic tasks are finding the data sources, retrieving data from those data sources and linking these data together.

5.4 DATA VIEW

The data in the system are basically of two kinds:

1. **Issue repository requirements** - contains all official require- ments of the project with further attributes. Can be used as a base ground for project repository information building.
2. **Text data** - all other data is expected to be text written in natural language. The only exception could be the source code reposi- tory with commit messages.

Table 5.3: Issue Properties interesting for prioritization

Name	Source	Description
Reporter	JIRA	User, who reported the issue.
Votes	JIRA	Votes collected from the users
Labels	JIRA Text mining	
Activity	Creation date Last Update Due date Code activity	Community activity statistics
Impact	x	x
Importance	x	x
Related users	Mail lists JIRA IM	Users, who discussed the issue.
E-mails	Mail Lists	List of related e-mails
IM discus- sions	IRC	List of related IM messages
Text de- scription	x	x
Priority	Priority Service	The computed priority of the issue

5.5 ARCHITECTURE

The architecture consists of a distributed Event Driven *Service oriented Architecture* (SOA). Based on several basic services: Data source ser- vices, Text mining service, Data linking service, Community service, Issues repository and Prioritization service, which will be described

Table 5.4: Community members Properties

Name	properties	Description
Date joined	First appear- ance of the member.	
Aliases	List of possible aliases	One user can use several aliases
Ranks	Community ranks of the user	Signifies the official member- ship
Communicati on Activity	Active First activity Last activity Response time Mood	The developer's involvement in the decision making and discussions process
Developer activity	LOC Issues resolved Last commit	Signifies the developer's im- portance
Other	Devices IP address Median active hours	Can be used for alias resolv- ing heuristics

in the following text. Due to the distributed nature, it can be properly scaled up with higher demand. Also the independence of the services means, that each service can be implemented with different technolo- gies. It is also possible to upgrade or change some service on the go. It is possible to implement high-availability or performance clustering or to possibly run the system in the cloud.

5.5.1 Services

1.Data Source Services are services, which are used for manag- ing the data input from different sources. If new data is found in the data source, then a message is emitted about this event.

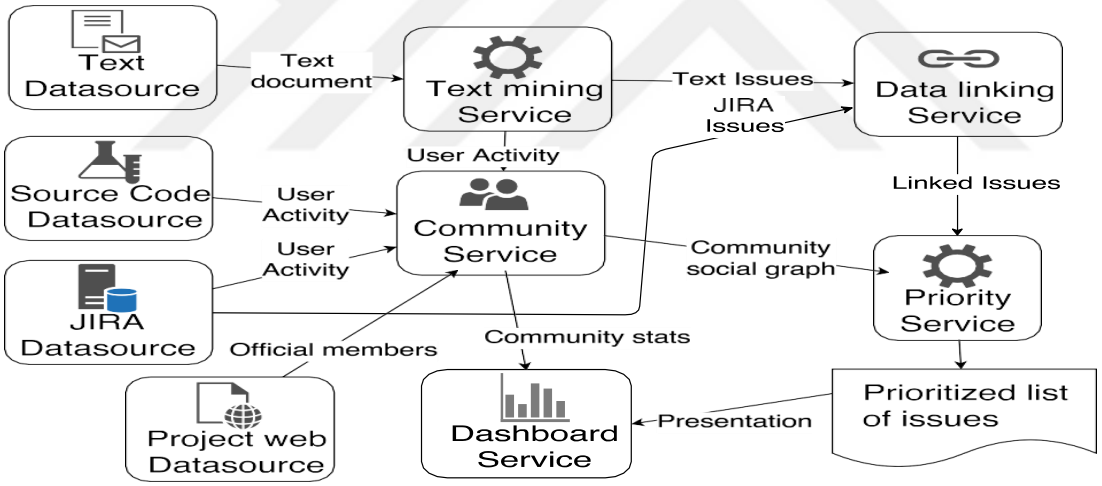

Figure 5.1: System diagram of basic services

The data source services serve only as the low level provider of the data. The transformations and processing are being made on higher level services.

A.Jira Service – manages the communication with the JIRA instance via REST API

B.Mail List Service – manages the project mail lists data

C.IRC Service – manages the IRC data source

D.Web Service – manages the other data sources

2. Data Linking Service connects the data from different data sour- ces

3. Prioritization Service computes the priorities of the issues and constructs the prioritized list of issues

4. Community service analyzes the social graph based on the com- munity members' activity. From the technical point, the service consists of a database of the community members, which is cre- ated and updated via processing the social events (see table 5.6 on page 34) emitted by the *Data Source* (DS) services (illustrated by the diagram 5.2). The social graph and user statistics is pro- vided to the Prioritization service for calculating the priorities.

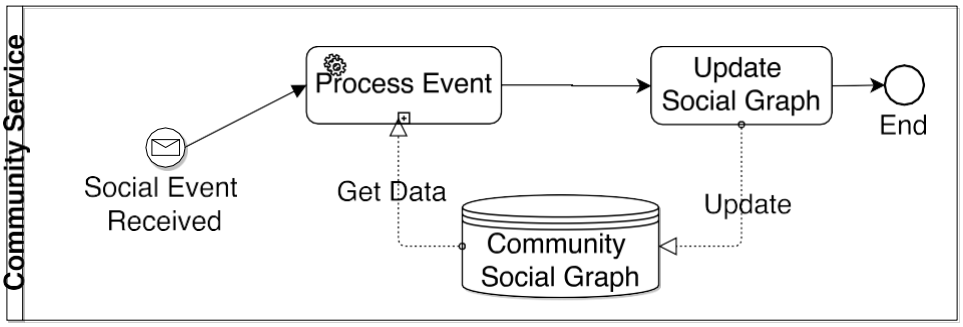

Figure 5.2: Community service: social event processing

5.6 INFORMATION LINKING

As we saw in the section 4.3.1 on page 16, earlier works assumed, that the mailing lists are the core part of the OSS development process. With our current analysis of the ASF projects with the most active committers, we saw a shift from the classic mailing lists communi- cation to issue tracking systems (namely JIRA). The mailing lists are still active and open to user's suggestions and questions. But the main part of the mailing communication traffic is devoted to mirroring the communication from the issue tracking system. If there is need for an text mining, then we propose to use cosine similarity method.

5.7 REQUIREMENT PRIORITIZATION AUTOMATION

This open development method is from the point of view of machine learning the main source for good training of the algorithms, because if the machine is being learned also by the rationale why the decisions
were made, we do expect it to advice smarter decisions compared to the elementary methods.

As we have seen in previous chapters, the concept of priority and prioritization is rather complex and in many cases not standardized nor well known to the team members. On the other hand, if the project is somehow managed regarding the issues priorities, still there might arise the scalability problems of the manual prioritization process, if the project grows somewhat big and thus still leads to the challenges for the project management to keep the project under control. We do propose utilizing the machine learning and text mining concepts to solve the problems of issue prioritization process. Firstly with this approach, we are able to bring some level of standardization to unify how the issues are being handled throughout the project. Secondly we are able to combine information from different sources together and process them, so the managers are not overloaded

with the pro- cess of analyzing the issues and at the same time we can deliver them the mined information in the form of prioritized list of issues. The last, but not least, benefit of this approach should be the scalability improvement over the manual processes, because of the highly auto- mated work and lower user interaction demand.

We are basing the prioritization model on Attlasian's Jira software,

which is getting to be de-facto standard in the requirements man- agement software and is being used among many OSS projects like Red hat's projects and ASF's projects, which will be used for the eval- uation part. To harmonize the used terms in the following text, we will refer to nowadays trendy "issue" term to encompass any of the terms like "feature", "bug", "issue" etc.[17] The priority is the result of the data mining of the multi dimensional available data (attributes) about the issues. We do propose a data model of useful attributes for the issue retrieval in the following section.

Literature offers us many insights into the intricacies and working of various requirement prioritization techniques. We have gathered certain valuable findings while working with these techniques as well. In this section, we shall briefly elaborate upon those techniques. Our experience has shown that for large projects with multi-objective requirements, AHP is a more preferred approach among the professionals. This technique yields statically very reliable results. The experience has also shown that the cost of conducting AHP is also marginal as compared to various other techniques. This technique however is not suitable in the situation where requirements are fast evolving and new requirements are being introduced at a much higher pace. The purely statistical nature of AHP (and it is true for other techniques described above as well) makes it difficult to generate a prioritization which accommodates these changes taking place. The technique due to it's highly time consuming nature also becomes an unfit solution for development models where several iterations take place (unless a sufficient time box is available for

prioritization process at each iteration). Since cumulative voting involves human insight apart from statistical techniques, we also experienced a more flexible and accommodating prioritization when handling changing and creeping requirements. The cost factor for cumulative voting becomes an inhibiting factor when we deal a project of several hundred requirements and a very tight budget. The element of bias was also visible in some of our experiments as experts inadvertently prioritized those requirements which they thought were more important from their perspective. We believe that AHP is more suitable for projects with medium number of requirements and waterfall or prototyping model. Cumulative counting on the other hand, can manage iterative development quite efficiently provided enough budget and expertise is provided. We have also observed that some kind of automation is required for both of these techniques. This automation can reduce the time requirement of AHP and make it more suitable for iterative development while it can also reduce the element of bias for cumulative counting and lend it more credibility. Numerical assignment technique was one of the most difficult to work with in our experience. Despite it being the most commonly used technique, we face almost insurmountable problems while working with numerical assignment. This technique was rendered useless when working in iterative environment. It was difficult to identify and gather all the stakeholders in each iteration, determining the exact status of their requirements (including all changes, creeps and incomplete) and then performing classification based prioritization. Our experience has shown us that numerical assignment is very unreliable when software is to be developed in iterations, has several stakeholders and fluidity of the requirements is very high. Some degree of success was achieved where the stakeholders are very few and highly oriented. Second problem while working with numerical assignment was the much greater degree of bias that was exhibited by stakeholder's when prioritizing than the bias we experienced in the case of cumulative voting.

Third problem that we experienced was that classification posed problems instead of solutions. It was because a large majority of requirements posed by various stakeholders were actually placed in the highest classification by their owners while requirements posed by other stakeholders were put in lower classifications as those were considered less important. Ranking was another prioritization mechanism which had very low potential in modern day development in its true sense. Top ten techniques is good at establishing a set of the most critical requirements. Our experience has shown that all these three techniques are very difficult to work with and can't meet the objectives of requirement prioritization in an optimal way. Theory W is a very valuable requirement prioritization technique. It has a two tier prioritization system which works within predefined limits. Stakeholders are given the opportunity to prioritize their own requirements which are then further studied and adjusted by experts before those are presented to all for negotiations. These negotiations last until we have asset of requirements in such a prioritized order that every stakeholder is a winner. We were able to get much better results by applying theory W than any other technique. The major problem that we faced while working with theory W was when requirements were fluid beyond certain degree. It was impossible to perform negotiations at each iteration. So the utility of this technique was somewhat diminished in iterative development with highly evolving and changing requirements. Planning game is also a better variant of numerical but the same problems persist (with somewhat less intensity). Wieger's method and requirement triage are relatively new entrants in the field of requirement prioritization. These techniques offer solutions to the problem of requirement prioritization which are more realistic and are more in sync with ground realities. These techniques are good in both linear and iterative process models. Our experimentations and observations have shown that AHP and cumulative voting are best existing techniques for linear and iterative models respectively.

5.8 TARGET PROJECT

The typical target project for this concept should be somewhat bigger with a large amount of issues. Since this thesis focuses on the open source world, we would like to aim at the open source projects which are being led in an open development model with meritocratic elements.

5.9 PROCESSED DATA

The processed data model is based on the basic build-in Jira Software issue attributes with some improvements for the future. The whole structure can be seen on the figure 5.3. The data attributes to be used by the machine learning can be altered to fit the specific needs of the particular project (and available data).

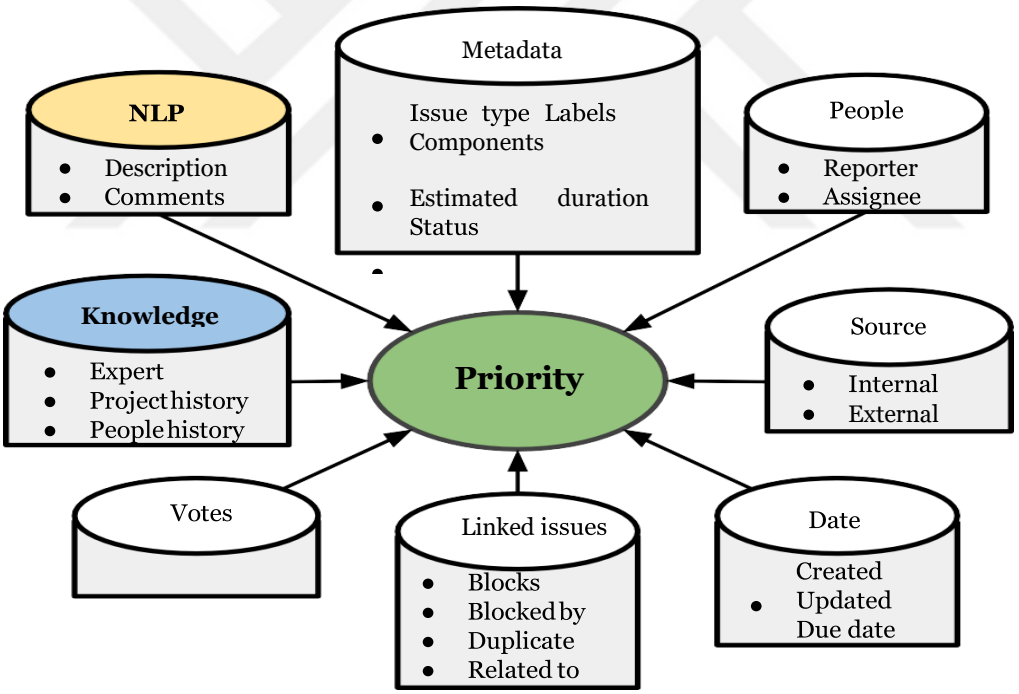

Figure 5.3: Priority depends on several issue attributes and other in- formation

5.9.1 NLP

This part is dedicated to the NLP and text classification. The text parts of the data like the description and comments can be mined

5.9.2 Knowledge

Expert knowledge and learned history (supervised learning) of the project to be used by the data mining.

5.9.3 Votes

Votes of the community members. Each member can vote for the is- sue to increase the priority. The vote has a weight, that signifies the member's rank in the community. The higher influence the member has, the more impact of his votes. Votes = $\sum rank_{weigth}; \forall user_{voted}$

5.9.4 Linked issues

The issue linking allows one to create a relation between two issues. In Jira Software, an issue might *be blocked by* another, *block* another. *duplicate* or simply *relate* to another issue.[16] If we have linked issues, then we can retrieve information about their relative priority from their relationship: *E*xample:

If Issue i_1 is blocked by issue i_2

Then: Issue i_1.priority $\leq$ issue i_2.priority

If: *Issue i_1 is blocked by issue i_2*

Then: Issue i_1.priority $\leq$ issue i_2.priority

Let's have Issues i_1 and i_2, then from the issue links, we can de- duce next information:

In reality, we would leave out one of the first two rules (*is blocked by* or *blocks*). The issue links would then form directed acyclic graph, so we wouldn't have to consider the recurring link cycles.

5.9.5 Dates

Also from the date attributes related to the issue can be learned some new information regarding the issue priority.
Due date As seen in the section **??**, the date might mean the due date (for example the planned release date of the issue). We do want to deliver on time. The priority of the issue should be increased as the dead- line is approaching, so there exist an inverse variation between the remaining time to resolve the issue and the priority for the remain- ing duration Δt:
date priority $= \frac{k}{\Delta t}$ for some well selected k (depends on the particular project)

5.9.6 Source

The original source of this issue might be useful to learn the impact of the issue.
Internal source – issue was created directly by a member of the inner circles of the project community, who has somewhat a high rank and is permitted to do so.

External source – this issue has emerged from the broader community. In this context – means from automatic text mining of the forums, mailing lists, app store reviews or QA sites to search for and acquire new feature or change requests or bugs reports. As seen for example in [19]

By default, the internal issue is of higher importance, so it has higher priority in comparison to the external issue. However in some cases it may be the opposite. For example, if the external issue concerns the whole community and is mentioned on many places and many people, then it has big impact and may surpass the internal issue by the means of priority

5.9.7 People

The people partly belongs to the internal source as the one of the peo- ple involved is the reporter, who reported the issue. The priority of the issue grows with the rank and influence of the reporter. Also some project would incorporate the assignee attribute – if an issue is as- signed to someone, then it can be considered more important than the issues not assigned to anyone yet.

Table 5.5: System services

JIRA Datasource	Communicates with the project JIRA
Mailing list Datasource	Collects e-mails from the lists
IRC Datasource	Collects messages from the IRC channels
Source code Datasource	Notifies the programming activity of the developers
Web Datasources	Custom services for searching external data related to the project
Text mining Service	Serves as a storage of the requirements and related data
Linking Service	Serves as a storage of the requirements and related data
Community Service	Tracks the social evolution of the community and provides the community social graph and user statistics
Issues Service	Serves as a storage of all the issues data
Prioritization Service	Computes the issues priorities based on the discovered information
Dashboard Service	Service for presenting the learned information

Table 5.6: Community Social Events

Name	Source	Description
Developer changed	Project Website	When the official community list is changed = Developer is added or removed.
Source Code change	Code Repository	Developer activity.
New message	Text mining	When user posts a new text message.

Table 5.7: Issue related Events

Name	Source	Description
New JIRA Issue	JIRA	When an issue is added to the JIRA database.
Update JIRA Issue	JIRA	When an issue is updated in JIRA.
New Text Issue	Data linking	When a new issue(which is not in JIRA yet) is discovered.
New data for issue	Data linking	When new text data is found for some existing issue.

6. PROTOTYPE AND EVALUATION

This chapter is devoted to the description of the prototype, the data, process and decisions, which led to the evaluated results.

6.1 PROTOTYPE

For the evaluation part, we chose to focus mainly on the base services of the system. We tried to cover the Data sources, text mining and requirements prioritization. The rest of the system components are higher level and depends on the previously mentioned, so that can be considered a future development of the prototype.

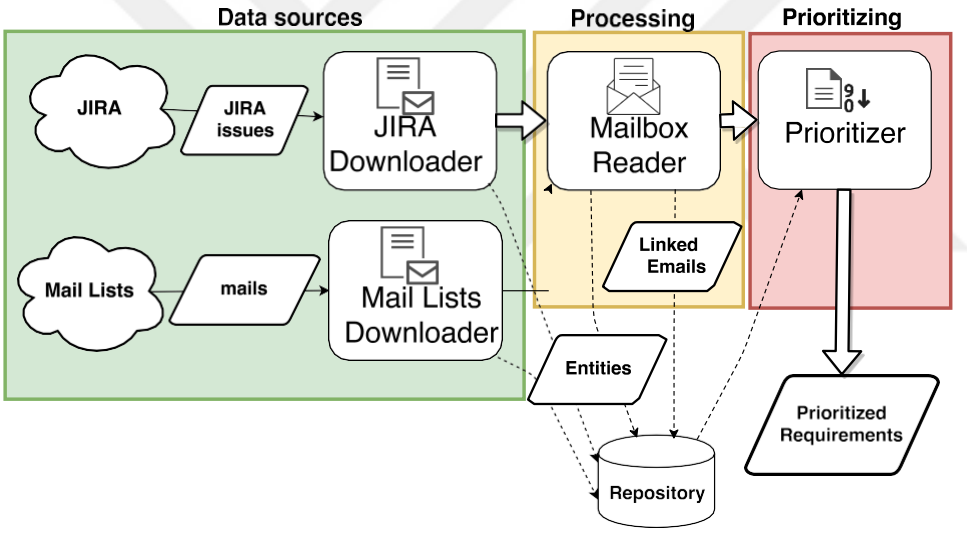

Figure 6.1: Components of the implemented prototype

6.1.1 Chosen Project

As we discussed in the previous chapter, we are interested in the projects developed with open development model. We chose to use the ASF projects. As the ASF incorporates the open development me- thod and has accessible

stance for the whole foundation). Also all details and discussion re- garding the project and decisions should be incorporated either in JIRA or forwarded to apropriate mailling list, even if it was discussed elsewhere (IRC or other method). So in the end, we should be able to easily get all the desired information.

6.1.2 Machine Learning

6.1.3 Data

The data downloaded and stored in the database repository provided by this prototype are basically from two sources:

- **JIRA**
  - Requirement
    - Requirement key - *unique*
    - Description
    - Summary
    - Issue type
    - Priority
    - Status
    - Resolution
    - Reporter
    - Created date
    - Assignee
    - Affect version

    - Fix version
    - Updated date
    - Resolved date
- User
    - Name - *unique*
    - Display name
    - E-mail
    - Active
- Comment
    - Requirement Key
    - Body
    - Author
    - Created date

- **Project Mail lists**

- id - *unique*
- Project
- Mail list
- Sender
- Date
- Subject
- Body

– References = referenced emails = email thread

6.1.4 Technology Stack

The prototype is written in python programming language, as it is good tool for fast prototyping. The Technology stack is following:

- **sqlite3** database with local file storage
- **python jira** module for REST client to the JIRA REST service
- **peewee** as a lightweight *Object relational mapping* (ORM) frame- work for storing and retrieving data to/from database
- **pandas** framework for priority prediction data management
- **numpy, sci py, sklearn** frameworks for priority prediction

6.1.5 JIRA Downloader

JIRA Downloader is basically a REST Client. The module is able to connect to a generic JIRA instance and download requirements data. The module is configured with next parameters:

- **URL** of the JIRA instance
- **Project** name (e.g. 'Hadoop')
- **Issue types** = enumeration of issue types, which should be down- loaded (e.g. 'Requirement')
- **Issue status** = enumeration of issue status, which should be

downloaded (e.g. 'Closed')

The basic operation consists of two steps:

1. **Requirement downloading**, which connects to the JIRA instance and tries to download all requirements according to the configuration.

2. **Comment downloading**, which is connecting to different REST endpoint and downloads text comments per-requirement.

6.1.6 Mail List Downloader

The mail list downloader module connects to an mbox archive site and downloads the required files into specified directory.

- **URL** of the mbox archives site
- **Project**
- **Mail lists**
- **Download dir**
- **Year, Month** start
- **Year, Month** stop

6.1.7 Mailbox Reader

The mailbox reader gets the file names from the Mail list downloader and operates in two steps:

1. **Save downloaded messages** into local database.
2. **Resolve references** after saving the requirements, the email th- reads can be resolved.

6.1.8 Prioritizer

The prioritizer module takes the previously created data repository and calculates the prediction of the priority.

After a discussion, we chose to predict the *priority* as a continuous value. That means, we will not use classification, but regression for predicting the priority value.

When we evaluated the possibility of using the classification al- gorithm, we conducted an experiment with classifying the priority based on the JIRA priority attribute. We chose to use the JIRA at- tributes: description, summary, reporter and issue type to predict the JIRA attribute priority (enumeration of values). The text data prepro- cessed with TF-IDF, classification with SVM algorithm and with tun- ing the parameters using grid search method on smaller batch of re- quirements (100), we set the TFIDF to remove english stop words, use (1,3) n-grams and tuned SVM alpha = $1e\ 3$ with the estimated pre- diction rate with mean accuracy around 70%.

However, we leaved the further evaluation of classification ap- proach in behalf of trying to predict continuous value of the priority with regression approach. The priority in our point of view is here de- fined as the index in the

ordering by resolving date, therefore, based on created date and other attributes, we are trying to predict the re- solved date. In other words, the priority depends on the creation date and the duration of the requirement. The shorter the duration is, the higher is the priority. The figure 6.2 on the following page illustrates this function. The ordering in the right "resolved Requirements" list means the priority of the requirements. In other words: the higher the priority value, the sooner the requirement gets resolved.

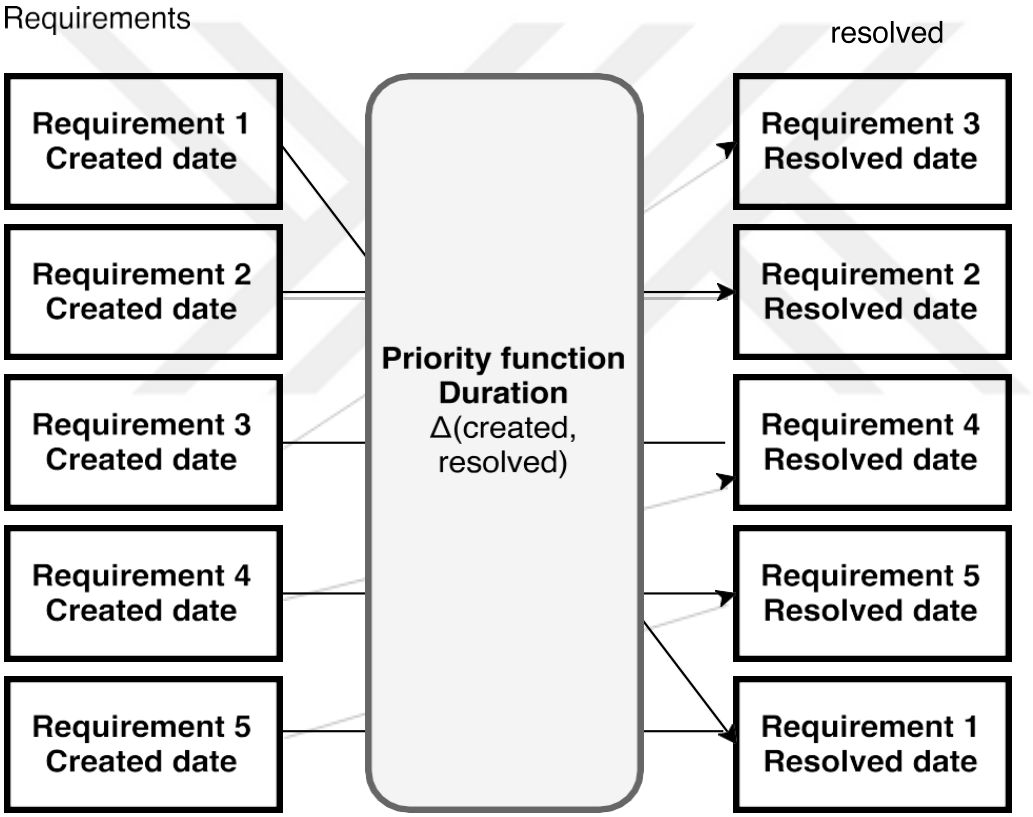

Figure 6.2: Priority function

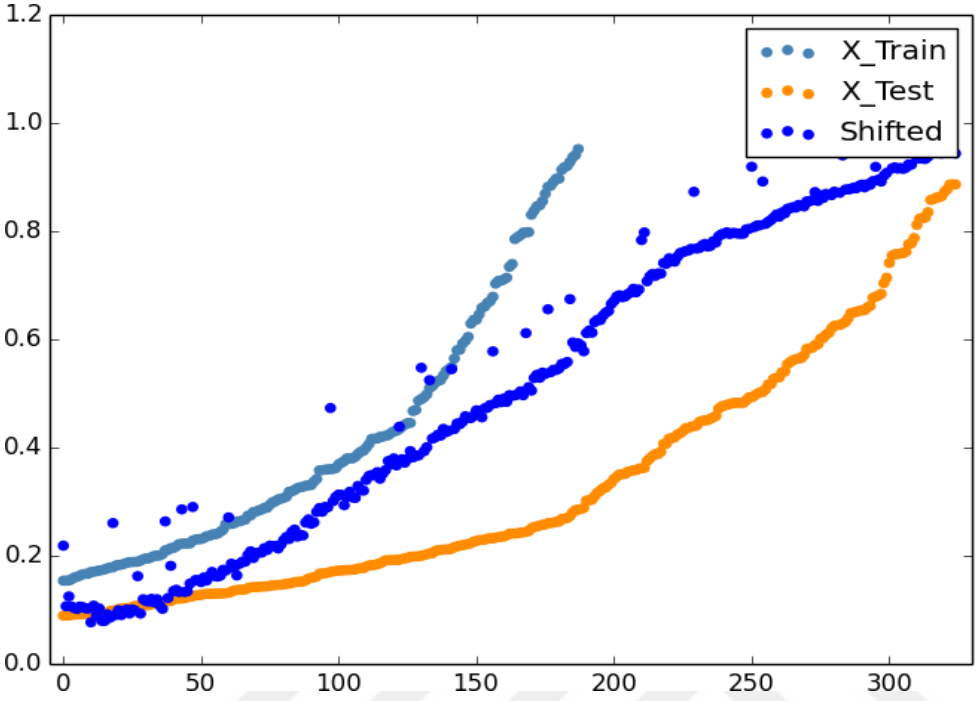

Figure 6.3: Requirement duration prediction using SVR method

6.2 Evaluation

We evaluated the proposed algorithm with data of the ASF Hadoop project. The dataset was obtained with our prototype directly from ASF JIRA. The evaluation process consisted of filtering the dataset, scaling the dataset and processing the text data with an TF-IDF vec- torization/transformation along with transforming also other feat- ures of the requirements. This mapping of data had been input into an SVM.

6.2.1 Evaluation Setup

- **JIRA** = ASF JIRA (https://issues.apache.org/jira)
- **Project** = Hadoop
- **Issue types** considered as belonging to Requirements = Improve- ment, New Feature, Task, Wish, Sub-task, Epic, Umbrella, Story, Technical task, Planned work, Request, Proposal
- **Issue status** = Resolved, Closed
- **Mails archives** = ASF Mail archives (http://mail-archives.apache.org/mod_mbox)
- **Mail lists** = common-commits, common-dev, common-issues, common-user, general, user

6.2.2 Dataset

The data retrieving took non-trivial amount of time, since the set of emails is not small and the processing from mbox file format took hours of time. Also the access to the JIRA REST service was problem- atic. Downloading large sets of data from JIRA wasn't easy, since after some data transit it threw time-outs or http session abortions, so the process of obtaining data costs much time. For this purpose, the pro- totype is equipped with a self recovery mechanism after unsuccessful data retrieving. The prototype remembers the last downloaded piece of data and starts at this position. The structure of data retrieved from the data sources are following:

Emails

The emails dataset consist of 382010 different messages spanning from January 2006 to the start of December 2017.

- **common-commits:** 75495 messages, commit messages with code changes
- **common-dev:** 98258 messages, general discussion messages, 90059 messages from JIRA
- **common-issues:** 142282 messages, notifications from jira
- **common-user:** 37691 messages, user related thread
- **general:** 7453 messages - general discussion
- **user:** 20831 messages, another user related thread, but more regarding the user issues with the using of the product, not with the development

After analysis of the emails dataset, we can say, that the meaning of mails which are not stored in JIRA is getting lower (92% of all emails in the mail list 'common-dev' origins in JIRA, so we assume that the role of information linking is nowadays insignificant.

6.2.3 Results

The setting of ML system was set the same as in the previous sec- tion.The grid search didn't yield much help, since there was no time left and not sufficient computing power to compute the right fitting parameters for larger number of requirements. The evaluation set- tings were the same with the proposed ones in previous section.The dataset of JIRA requirements was divided to 9 folds of 500 require- ments. The evaluation consisted of 8 iterations run. Each iteration had growing training dataset by one fold as illustrated on figure A.8 on page 55.

Table 6.1: Automated requirement prioritization prototype evalua- tion results

Iteration	Training	SVM Score	Kendall	Kendall *p* value
1	500	-1.704639	-0.000375	0.995012
2	1000	-4.401022	-0.357508	0.000001
3	1500	-5.618613	0.019622	0.804647
4	2000	-1.515781	-0.023564	0.736550
5	2500	-0.843487	**0.294505**	0.000008
6	3000	-3.773701	-0.128141	0.025225
7	3500	-6.688573	-0.028749	0.609166
8	4000	-9.269804	0.033466	0.542008

The results can be seen in the summary table of results 6.1 The proposed solution is in the state of prototype. As we can see from the evaluation, it doesn’t behave well in general terms of requirement prioritization yet and is not applicable to gen- eral data, but in some cases (iteration 5) can predict the data with some accuracy. The worsening of the prediction with latter iterations can be caused either by some bug in the prototype code or there is also possibility, that the project evolved in time to much, that the predic- tion is not directly possible without further transforming the source data.

7. CONCLUSIONS

The work presented here can be extended in many ways in the fu- ture work. In this thesis, we address the problem of using machine learning approaches for requirements prioritization in the context of OSS. We analyzed the current state of the art in this research field and found out, that there is not much research being done the field of requirements prioritization in the context of the OSS projects.

Given the results of the previous research work, we used a synthe- sis way to combine different approaches and algorithms to propose a system, which would solve the problem of requirements prioritiza- tion in the context of OSS projects. The second step was to implement a prototype of such system, which would use some features of the proposed solution to predict the requirements priority. The next step was the evaluation of such design in the context of real open source project and adapting the original design to real world scenarios. So the first contribution is adaptation of the prime approaches posted by former researchers to the context od OSS projects.

The next contribution was the implementation of prototype. At first we conducted a short evaluation of priority classification, but shortly afterwards switched to regression priority prediction. The pro- totype is not directly applicable to general data, but can be used as a base ground for further research in this context. Also the prototype is able to create data repositories of software projects, which use JIRA and mail lists.

The next contribution was the creation of repository consisting of 10 year project life span of ASF project Hadoop containing its require- ments and emails data. This database was created directly with the prototype.

Evaluation results

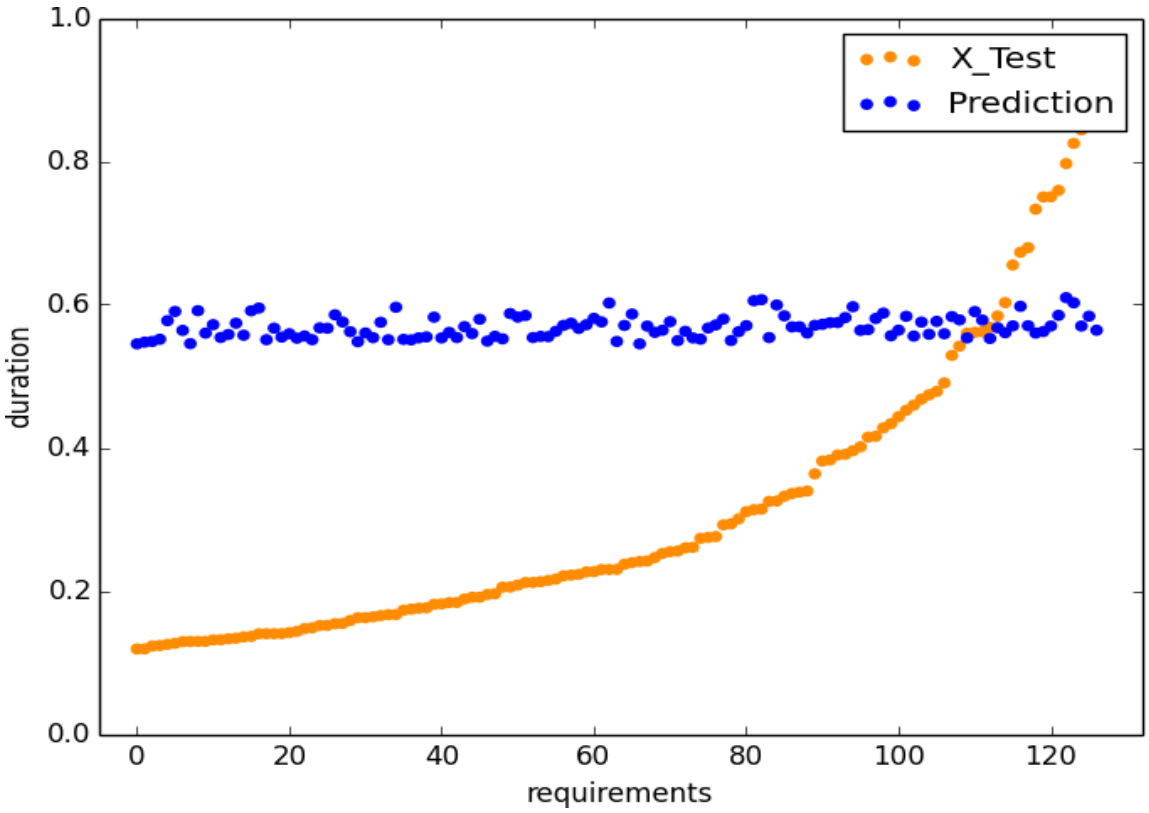

Figure 7.1: Evaluation, iteration #1, 500 requirements

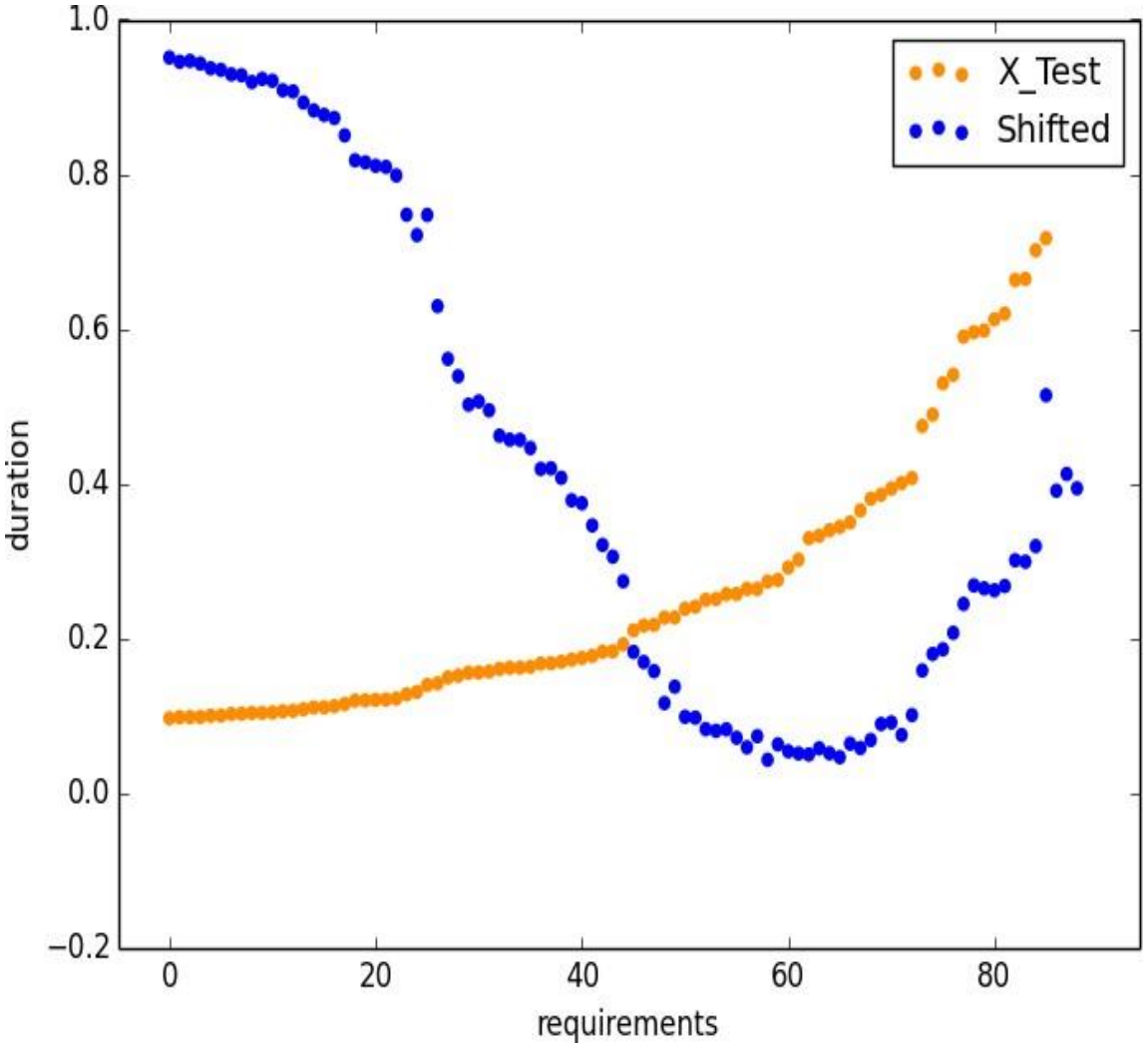

Figure 7.2: Evaluation, iteration #2, 1000

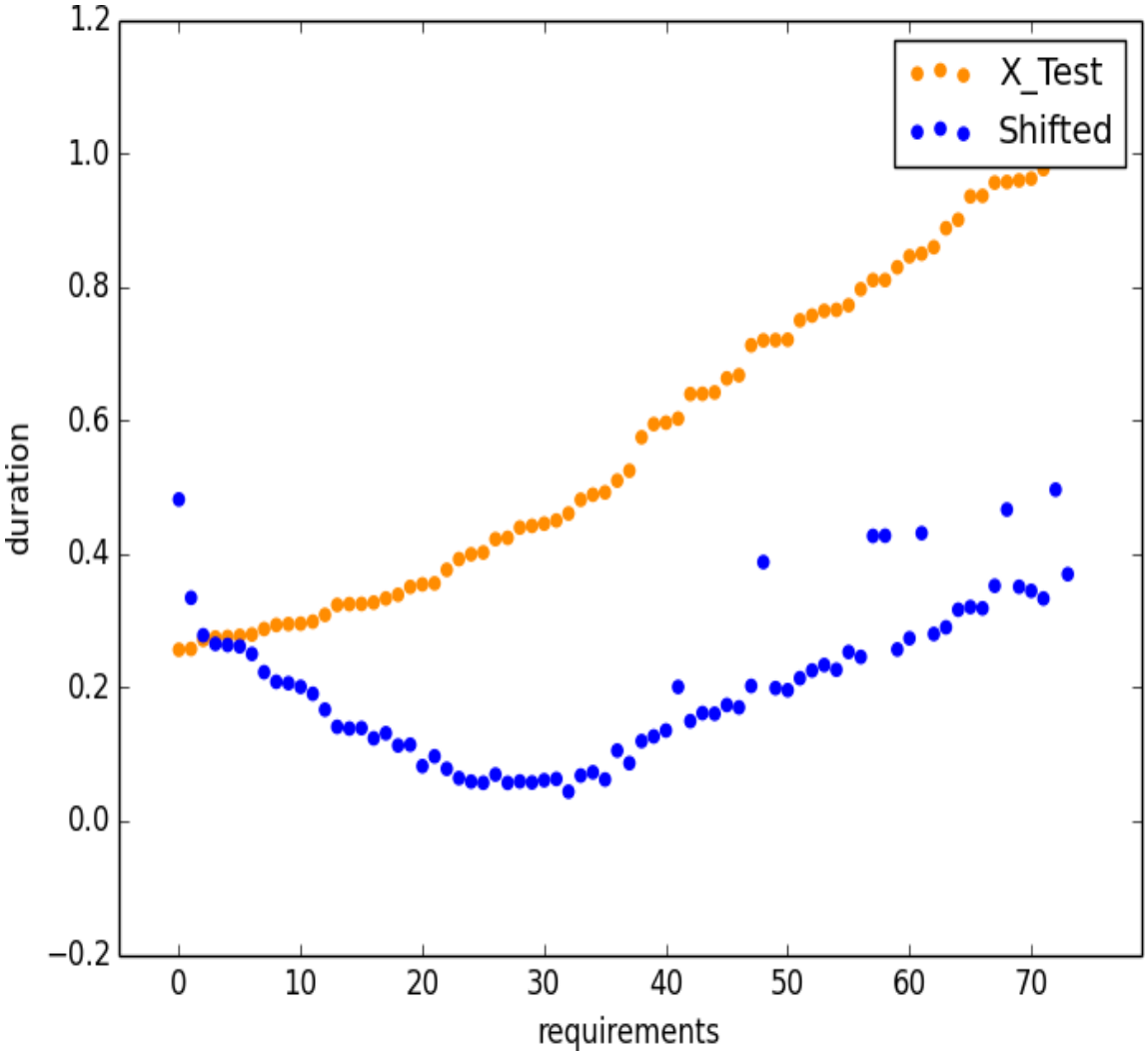

Figure 7.3: Evaluation, iteration #3, 1500 requirements

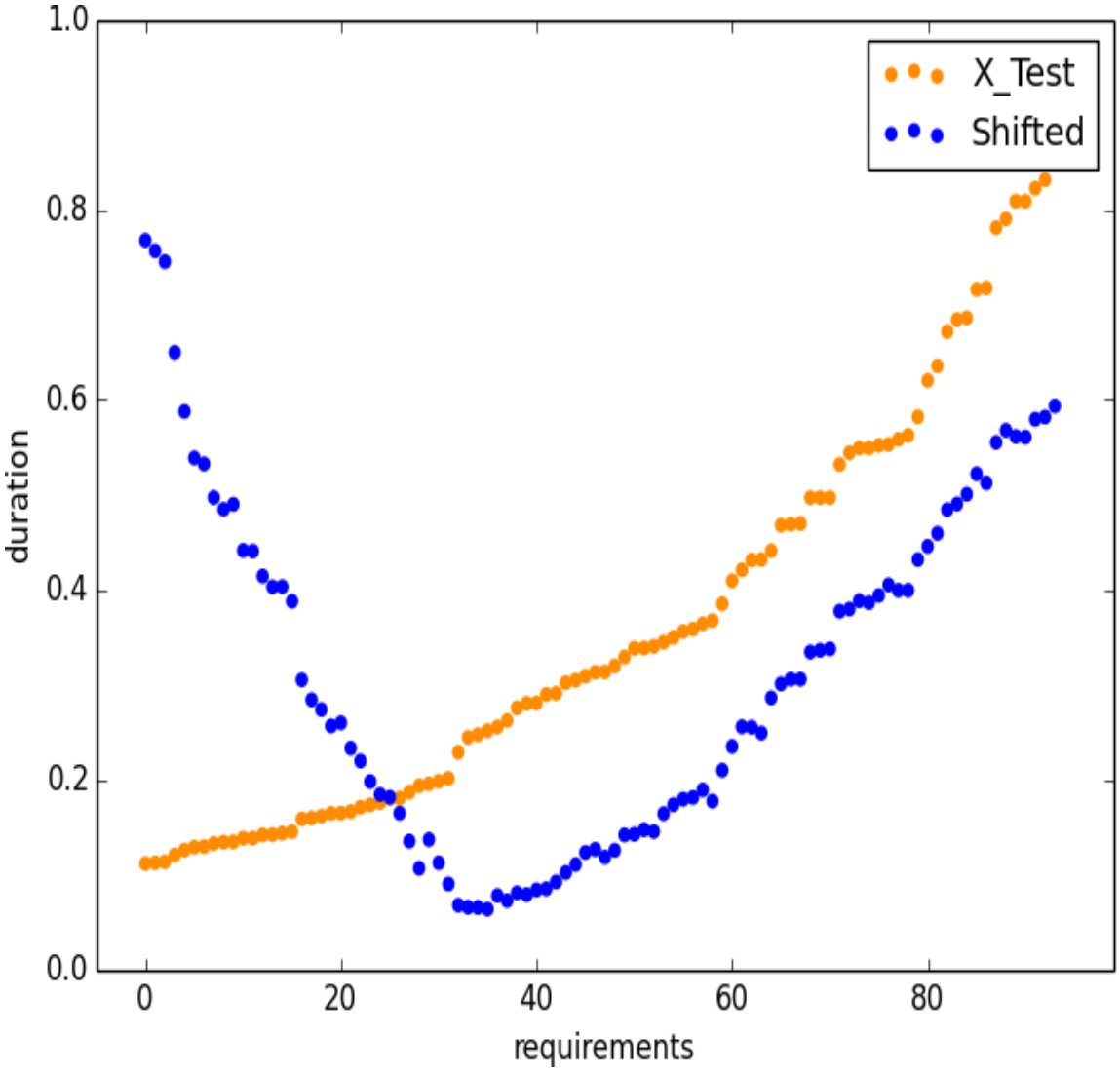

Figure 7.4 : Evaluation, iteration #4, 2000 requirements

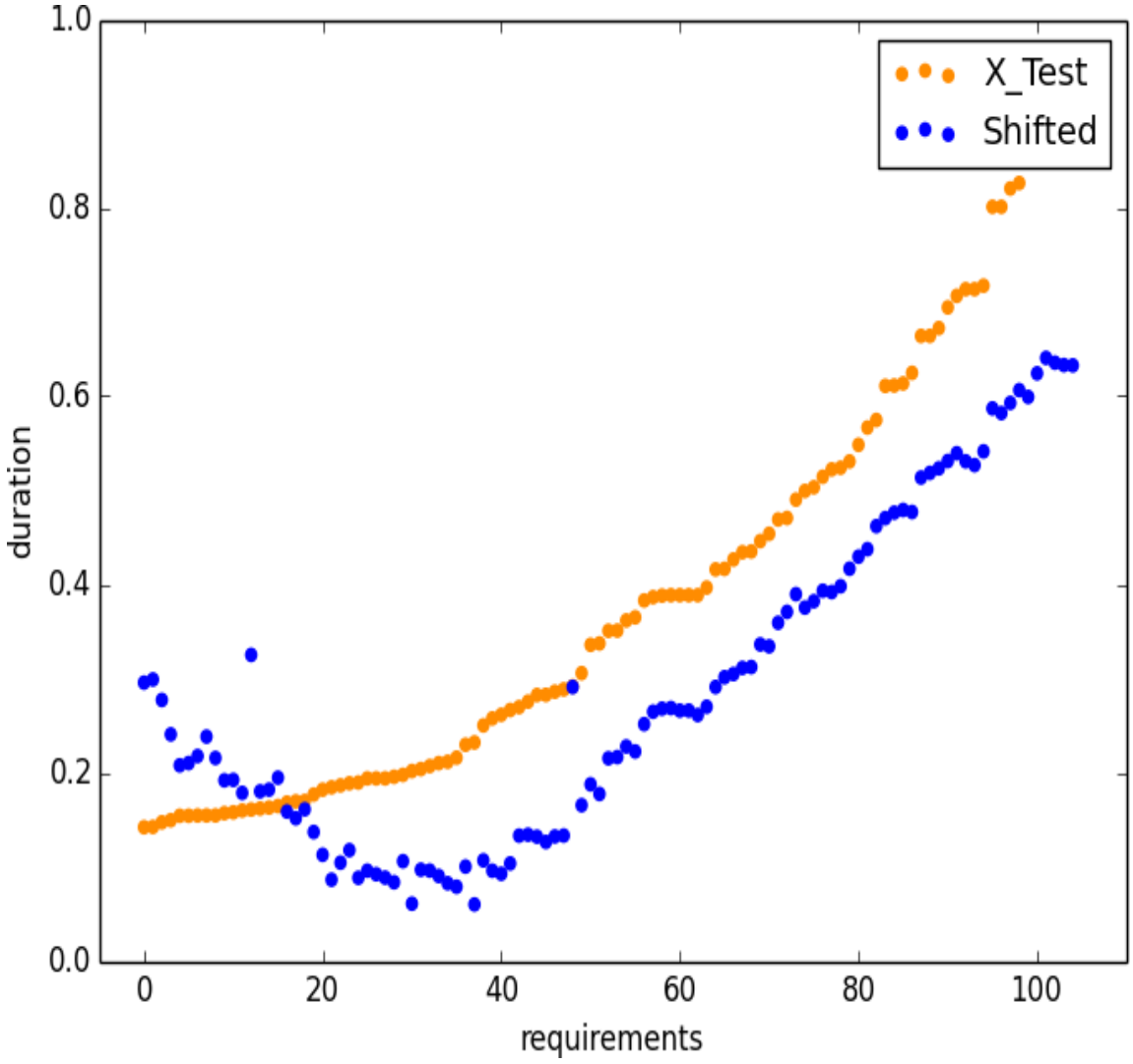

Figure 7.5 : Evaluation, iteration #5, 2500 requirements

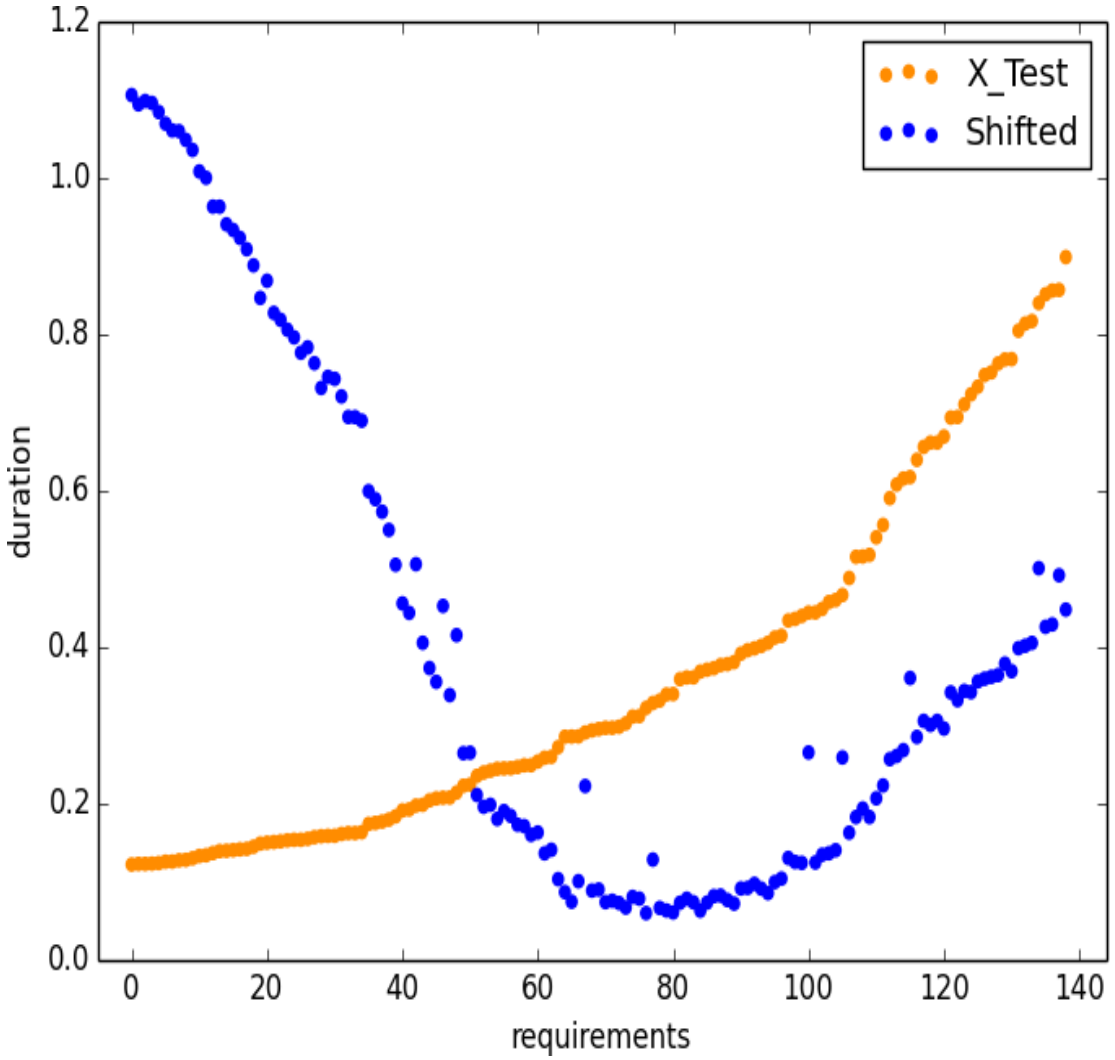

Figure 7.6 : Evaluation, iteration #6, 3000 requirements

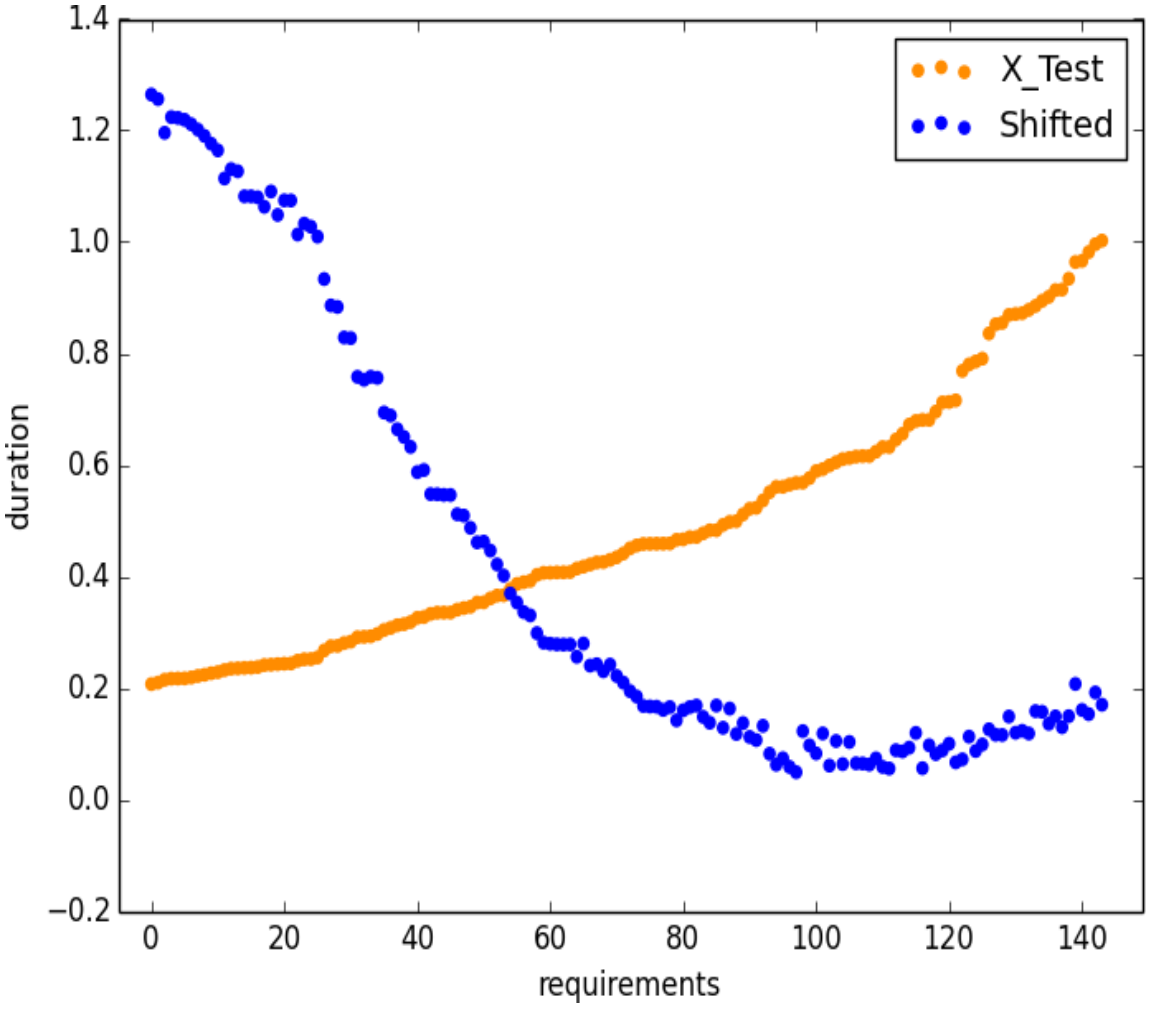

Figure 7.7 : Evaluation, iteration #7, 3500 requirements

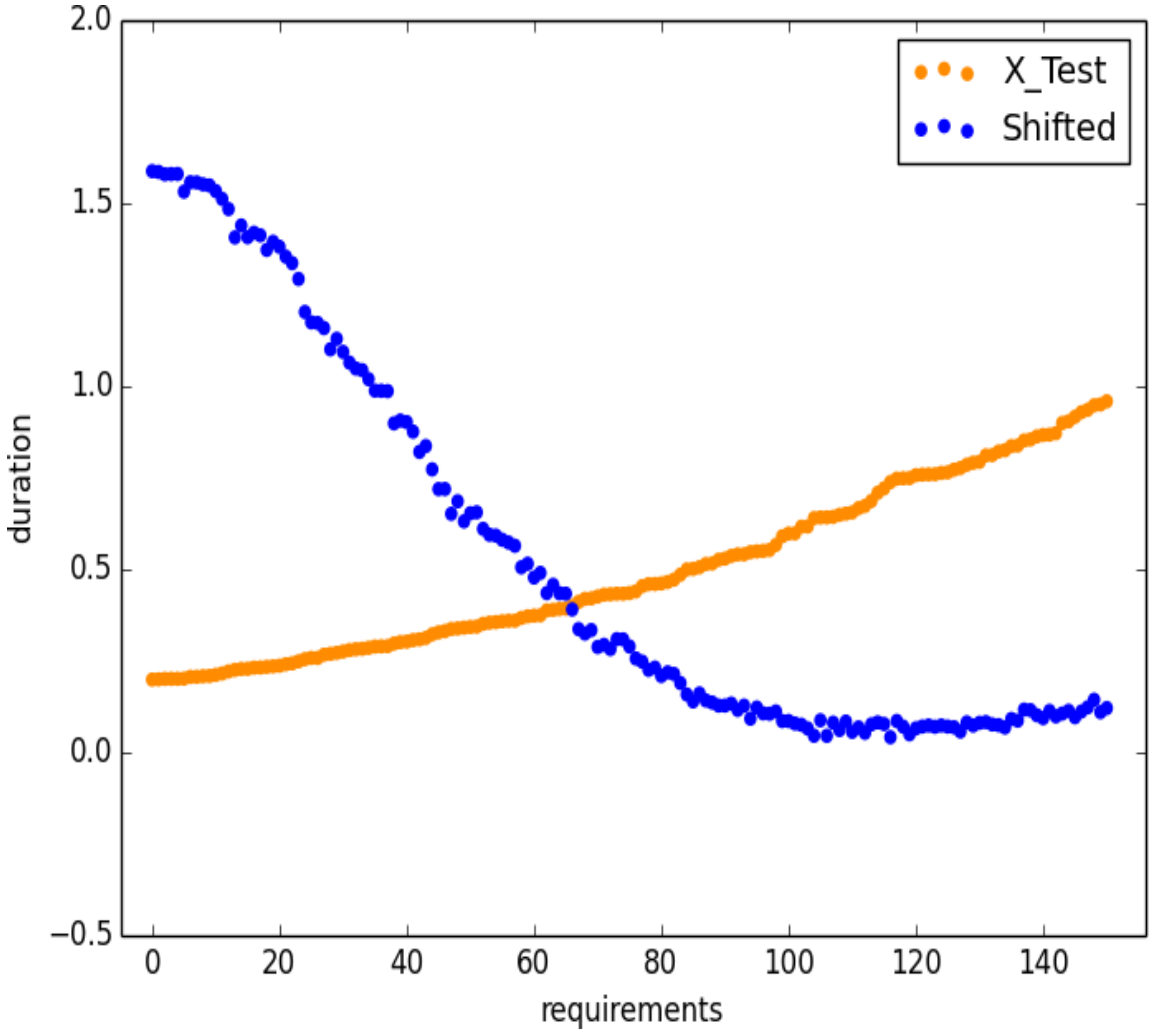

Figure 7.8 : Evaluation, iteration #8, 4000 requirements

REFERENCES

[1] LEHOTA, Laura, Kauppinen MARJO, Kaulaja SARI, Bomar- ius FRANK and Iida HALIMU. *Requirements Prioritization Chal- lenges in Practice.* Product Focused Software Process Improve- ment. 2004. Berlin: Springer Berlin Heidelberg, 2005, p. 497-508. ISBN 9783540246596.

[2] BABAR, Muhammad Imran, Muhammad RAMZAN a Shah- baz A. K. GHAYYUR. *Challenges and future trends in software re- quirements prioritization.* In: International Conference on Com- puter Networks and Information Technology [online]. IEEE, 2011, p. 319-324 [cit. 2016-12-11]. DOI: 10.1109/ICCNIT.2011.6020888. ISBN 978-1-61284-940-9. Available: http://ieeexplore.ieee. org/document/6020888/

[3] GUZZI, Anja, Alberto BACCHELLI, Michele LANZA, Martin PINZGER a Arie VAN DEURSEN. *Communication in open source software development mailing lists.* In: 2013 10th Working Confer- ence on Mining Software Repositories (MSR) [online]. IEEE, 2013, p. 277-286 [cit. 2016-12-11]. DOI: 10.1109/MSR.2013.6624039. ISBN 978-1-4673-2936-1. Available: http://ieeexplore.ieee. org/document/6624039/

[4] PERGHER, Massimiliano a Bruno ROSSI. *Requirements priori- tization in software engineering: A systematic mapping study.* In: 2013 3rd International Workshop on Empirical Requirements En- gineering (EmpiRE)[online]. IEEE, 2013, p. 40-44 [cit. 2016-12- 09]. DOI: 10.1109/EmpiRE.2013.6615215. ISBN 978-1-4799-1011.

[5] PERINI, Anna, SUSI Angelo, and AVESANI Paolo, *A Machine Learning Approach to Software Requirements Prioritization*, IEEE Transactions on

Software Engineering, vol. 39, no. 4, p. 445–461, 2013.

[6] ACHIMUGU, Philip, Ali SELAMAT, Roliana IBRAHIM and Mohd Naz'ri MAHRIN. *A systematic literature review of soft- ware requirements prioritization research.* Information and Software

[7] ACHIMUGU, Philip, SELAMAT, Ali, and IBRAHIM Roliana, *A Web-Based Multi-Criteria Decision Making Tool for Software Require- ments Prioritization*, D. Hwang et al. (Eds.): ICCCI 2014, LNAI 8733, pp. 444–453, 2014.

[8] BABAR, Muhammad Imran, Masitah GHAZALI, Dayang N.A. JAWAWI, Siti Maryam SHAMSUDDIN a Noraini IBRAHIM. PHandler: *An expert system for a scalable software requirements prioritization process. Knowledge-Based Systems* [online]. 2015, 84, 179-202 [cit. 2016-12-07]. DOI: 10.1016/j.knosys.2015.04.010. ISSN09507051.Availablehttp://linkinghub.elsevier.com/retrieve/pii/S0950705115001483

[9] ACHIMUGU, Philip, Ali SELAMAT a Roliana IBRAHIM. *Re- proTizer: A Fully Implemented Software Requirements Prioritiza- tion Tool* [online]. p. 80 [cit. 2016-12-07]. DOI: 10.1007/978-3- 662-49619-0_5. Available: http://link.springer.com/10.1007/ 978-3-662-49619-0_5

[10] IAO, Ming, Gang YIN, Tao WANG, Cheng YANG a Mengwen CHEN. *Requirement Acquisition from Social Q&A Sites* [online]. p. 64 [cit. 2016-12-10]. DOI: 10.1007/978-3-662-48634-4_5. Available: http://link.springer.com/10.1007/978-3-662-48634-4_5

[11] HAN, Jiawei, Micheline KAMBER a Jian PEI. *Data mining: con- cepts and techniques*. 3rd ed. Boston: Elsevier, 2012, xxxv, 703 p. Morgan Kaufmann

series in data management systems. ISBN 978-0-12-381479-1.

[12] Jonna, KORHONEN a Riku SUOMELA.*Open Source Software Development: Requirements Elicitation Methods for Open Source Software Systems.* 2015. University of Oulu. Available: https://wiki.oulu.fi/download/attachments/58197330/ossd_2015_alkhanji_korhonen_suomela.pdf?version=1& modificationDate=1448956481000&api=v2

[13] LAURENT Paula and CLELAND-HUANG Jane, Martin a PATRICK HEYMANS (EDS.). Lessons Learned from Open Source Projects for Facilitating Online Requirements Processes, Re- quirements engineering: foundation for software quality 15th International Working Conference, REFSQ 2009 Amsterdam, The Netherlands, June 8-9, 2009: proceedings. [Online] [cit. 2017-12-13]. pages 240-255, Berlin: SpringerLink, 2009. ISBN 9783642020506.

[14] GARDLER, Ross and Gabriel HANGANU. Governance Mod- els. *OSS Watch* [online]. 2013 [cit. 2016-04-26]. Available: http: //oss-watch.ac.uk/resources/governancemodels

[15] A. Mockus, R. T. Fielding, and J. D. Herbsleb. A case study of open source software development: the apache server. In Proc. of ICSE'00, pages 263–272, 2000.

[16] Linking issues. *Attlassian documentation* [online]. [cit. 2016- 05-04]. Available: https://confluence.atlassian.com/ jirasoftwarecloud/linking-issues-776997756.html

[17] What is an Issue. *Attlasian documentation* [online]. [cit. 2016-05- 05]. Available: https://confluence.atlassian.com/jira064/ what-is-an-issue-

720416138.html

[18] The ultimate guide for software development teams us- ing Kanban. *Kanbanize* [online]. [cit. 2016-05-05]. Available: https://kanbanize.com/kanban-resources/case-studies/ kanban-for-software-development-teams/

[19] IAO, Ming, Gang YIN, Tao WANG, Cheng YANG a Mengwen CHEN. *Requirement Acquisition from Social Q&A Sites* [online]. p. 64 [cit. 2016-05-05]. DOI: 10.1007978-3-662-48634-4_5. Available: http://link.springer.com/10.1007/978-3-662-48634-4_5

[20] ANDERSON, Paul. Meritocrats, Cluebats And The Open De- velopment Method: An Interview With Justin Erenkrantz. *OSS Watch* [online]. Intelligent content, 2012 [cit. 2016-04-26]. Avail- able: http://oss-watch.ac.uk/resources/erenkrantz

[21] Daneva, M. and van der Veen, E. and Amrit, C. and Ghaisas, S. and Sikkel, K. and Kumar, Ramesh and Ajmeri, N. and Ram- teerthkar, U. and Wieringa, R.J. (2013) *Agile requirements prioritiza- tion in large-scale outsourced system projects: an empirical study.* Jour- nal of systems and software, 86 (5). 1333 -1353. ISSN 0164-1212

[22] HINTERBERGER, Hans, Josep DOMINGO-FERRER, Vipul KASHYAP, et al. Text Classification. *Encyclopedia of Database Systems* [online]. Boston, MA: Springer US, 2009, p. 3044 [cit. 2016-12-13]. DOI: 10.1007/978-0-387-39940-9_3791. ISBN 978-0-387-35544-3. Available: http://www.springerlink.com/index/10.1007/978-0-387-39940-9_3791

[23] MANNING, Christopher D., Prabhakar. RAGHAVAN and Hin- rich. SCHÜTZE. *Introduction to information retrieval*. New York: Cambridge University Press, 2008. ISBN 0521865719.

[24] YANG, Yiming. An Evaluation of Statistical Approaches to Text Categorization. *Information Retrieval* [online]. 1(1/2), 69-90 [cit. 2016-12-13]. DOI: 10.1023/A:1009982220290. ISSN 13864564. Available: http://link.springer.com/10.1023/A: 1009982220290

[25] CROWSTON, Kevin a Ivan SHAMSHURIN. *Core-Periphery Com- munication and the Success of Free/Libre Open Source Software Projects* [online]. p. 45 [cit. 2016-12-15]. DOI: 10.1007/978-3- 319-39225-7_4. Available: http://link.springer.com/10.1007/ 978-3-319-39225-7_4

[26] HANNEMANN, Anna, Kristjan LIIVA a Ralf KLAMMA. *Navi- gation Support in Evolving Open-Source Communities by a Web-Based Dashboard* [online]. p. 11 [cit. 2016-12-15]. DOI: 10.1007/978-3- 642-55128-4_2. Available: http://link.springer.com/10.1007/ 978-3-642-55128-4_2

[27] Process. *Apache Software Foundation* [online]. [cit. 2016- 12-20]. Available: https://www.apache.org/foundation/ how-it-works.html

[28] Sklearn.svm.SVR. *Scikit learn* [online]. 2017 [cit. 2017-12- 12]. Available: http://scikit-learn.org/stable/modules/ generated/sklearn.svm.SVR.html

[29] SMOLA Alex J. and SCHOLKOPF Bernhard . A tutorial on sup- port vector regression *Statistics and Computing 14* 2004 [online]. p. 199-222 [cit. 2017-12-12]. Available: http://scikit-learn.org/ stable/modules/generated/sklearn.svm.SVR.html

[30] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM* A library for Support Vector Machines [online]. 2013 [cit. 2017-12-12]. Avail- able: http://scikit-learn.org/stable/modules/generated/ sklearn.svm.SVR.html

[31] Tuning the hyper-parameters of an estimator. *Scikit learn* [on- line]. 2017 [cit. 2017-12-12]. Available: http://scikit-learn. org/stable/modules/grid_search.html#grid-search

[32] NOETHER Gottfried E. , *Elements of Nonparametric Statistics*", New York: John Wiley & Sons, 1967