

Universidade Federal do Pampa

Claudio Davi de Souza

**Role Identification Platform, An Automated
Tool To Identify Roles Inside Self-Organizing
Software Development Teams**

Alegrete

2016

Claudio Davi de Souza

Role Identification Platform, An Automated Tool To Identify Roles Inside Self-Organizing Software Development Teams

Trabalho de Conclusão de Curso apresentado
ao Curso de Graduação em Engenharia de
Software da Universidade Federal do Pampa
como requisito parcial para a obtenção do tí-
tulo de Bacharel em Engenharia de Software.

Supervisor: Pr. Dr. Marcelo Resende Thielo

Alegrete

2016

Claudio Davi de Souza

Role Identification Platform, An Automated Tool To Identify Roles Inside Self-Organizing Software Development Teams

Trabalho de Conclusão de Curso apresentado
ao Curso de Graduação em Engenharia de
Software da Universidade Federal do Pampa
como requisito parcial para a obtenção do tí-
tulo de Bacharel em Engenharia de Software.

Trabalho de Conclusão de Curso defendido e aprovado em de de

Banca examinadora:

Pr. Dr. Marcelo Resende Thielo
Orientador

Pr. Me. Jean Felipe Patikowski
Cheiran

Pr. Me. Eliezer Soares Flores

Acknowledgements

I would like to give a very special thanks to Marcelo Thielo, for helping me through all of this work, for coping with my paranoia and motivating me during the process. None of this would be even close to be achieved if it weren't for him. Thank you for being there to help me through the hardships, thank you for our good times, conversations, laughter during our meetings, ideas and encouraging words, I cannot possibly thank you enough, *qatlho*'.

Also thank you to all of my friends and colleagues that saw me disappear for days working on this project and managed to get me out sometimes. For making me laugh when I was stressed out and for still being friendly even when I was not being a good friend.

"Would you tell me, please, which way I ought to go from here?"
"That depends a good deal on where you want to get to."
(Alice in Wonderland - Lewis Carroll)

Resumo

A Engenharia de Software sofreu mudanças drásticas desde a introdução das metodologias ágeis. Gerentes de projeto tem que lidar com a escolha, seleção e manutenção dos membros de um time de desenvolvimento ágil. Muitas dessas atividades requerem um alto grau interpretativo dos dados demonstrados pelas ferramentas de gestão de projetos de software como, por exemplo a identificação de papéis informais dentro de um time, e se há discrepância entre atividades de um mesmo membro. Técnicas de aprendizado de máquina podem auxiliar o gerente de projeto na tomada de decisão. Neste trabalho, é demonstrada a necessidade de subsidiar o gerente de projeto com mais informações, sendo apresentados também trabalhos já realizados na área e finalmente é apresentada uma solução que utiliza agrupamento de dados para designar e identificar os papéis dentro de um time de desenvolvimento ágil auto-organizável.

Palavras-chave: Gerenciamento de Projetos, Aprendizado de Máquina, Clustering, Papéis, Times de desenvolvimento de Software

Abstract

Software Engineering deeply changed with the widespread use of Agile Methodologies. Project managers have to deal with the assignment and maintenance of an agile software development team. Several of the manager activities require a very well established interpretative skill to make better decisions based on the information provided by the software management tool. As an example, we can cite the identification of informal roles inside a software team and the discrepancy of expected activities between the team members. Machine Learning techniques can help the project manager dealing with everyday decisions. In this work, we assess the need to support the Project Manager with more information. Related works and solutions in the field are also reviewed and finally a solution using Clustering techniques to assign and identify the roles played inside a self-organizing software development team using agile is presented.

Key-words: Project Management, Machine Learning, Clustering, Roles, Software Teams

List of Figures

Figure 1 – Scrum Summary	23
Figure 2 – Clustering Example	26
Figure 3 – Facilitator Roles in self organizing teams (HODA; NOBLE; MAR-SHALL, 2010).	29
Figure 4 – Diagram showing the proposed solution being used with Scrum	33
Figure 5 – Component diagram illustrating the architecture model designed for the platform	41
Figure 6 – Base domain model used to help visualize the relationship between classes	42
Figure 7 – Final user interface mockup	43
Figure 8 – Final user report settings interface	44
Figure 9 – Solved issue on JIRA interface	45
Figure 10 – Clustering Options	45
Figure 11 – Report Example with sample data	46
Figure 12 – Report example with 2 clusters	47
Figure 13 – Report Example with 3 clusters	48
Figure 14 – New visualization model proposal	49

List of Tables

Table 1 – Approximate Web Developer profile collected from 4 different companies websites	35
Table 2 – User Stories	40
Table 3 – Approximate Web Developer Profile collected from 4 different companies websites	57
Table 4 – Approximate Tester Profile collected from 4 different companies websites	57
Table 5 – Approximate Software Engineer profiles collected from 3 different tech companies websites	58

Contents

1	INTRODUCTION	19
1.1	Motivation	19
1.2	Objectives	19
1.2.1	Goals	20
1.3	Structure	20
2	BACKGROUND	21
2.1	Agile Methods	21
2.1.1	Scrum	22
2.1.1.1	Scrum Summary	22
2.1.1.2	Scrum Team	22
2.2	Software Project Management	24
2.3	Machine Learning	25
2.3.1	Data Clustering	26
2.3.2	Principal Components Analysis	27
3	RELATED WORK	29
4	PROPOSED SOLUTION	33
4.1	Presentation	33
4.1.1	Design and Implementation	33
4.2	Methodology	34
4.2.1	Collect Evidence	36
4.2.2	Analyzing Answers	36
5	ROLE IDENTIFICATION PLATFORM	39
5.1	Design	39
5.1.1	User Stories	39
5.1.2	Architecture and Design	39
5.1.2.1	Architecture	39
5.1.2.2	Domain	40
5.1.3	JIRA integration	40
5.1.3.1	Notation	41
5.1.4	User Experience	43
5.2	Usage	43
5.2.1	Installation	43

5.2.2	Using JIRA with RIP	44
5.2.3	Using RIP	44
5.2.4	Result Analysis	46
5.3	Future Works	48
6	CONCLUSION	51
	BIBLIOGRAPHY	53
	APPENDIX	55
	APPENDIX A – PROFILES COLLECTED	57
	APPENDIX B – FORMATTED DATASET	59
	APPENDIX C – SAMPLE DATASET	61

1 Introduction

1.1 Motivation

Agile Methodologies are the most common software development management processes in use today. It introduces concepts like self-organizing teams, which brought along one important aspect of project management, that can be described with one question: "How can I assign roles and verify them in a software team?".

Most of the research being done in the field covers several soft-skills that must be present on a team, but very few of them tackle the technical skill set required for each role.

Several tools were developed to provide the Project Manager with all the crucial information about the project being managed. Although, he/she still has to interpret it and make decisions that may lead to the success or downfall of the whole endeavor. This information however, is entirely subjective to the Project Manager interpretative skills and experience. Moreover, the current available tools for project management lack the continuous verifications of the team's roles and work. It may clarify what is being done by whoever is doing it, but the variation of activities through time is not something that the technique is able to cover yet.

The decision making process inside a software development project is intense and sometimes excruciating. Even professionals with many years of experience still need the right information from the right tools to achieve the expected level of certainty, before making a decision.

When a team is assigned to a project, the project manager has to understand and trust that each role is being properly fulfilled. Software projects are mostly about the teams than anything else. However, sometimes teams are hard to understand. They are often composed by experts and ideally each one of them are at least equally, and sometimes even more skilled than the manager at their jobs. How do we identify those people? How do we know that what they are doing is exactly what they are supposed to be doing?

1.2 Objectives

In this work, we intend to cover the relation between the technical skills and activities of the team. Using information already made available by project management tools, we intend to classify and make continuous verification of the roles played inside

a software development team. Therefore saving work hours and energy of the project manager and reducing risks for the project.

1.2.1 Goals

- Identify the main roles inside an agile software development team.
- Identify the main tasks performed by each role.
- Design and implement a software capable to identify roles and assign team members to them based on activities performed by its members.
- Use Machine Learning algorithms and tools to solve real world problems.

1.3 Structure

In Background ([chapter 2](#)) we will cover some of the basis of this work. In the Agile section ([section 2.1](#)) we will make an overview about agile and will go in detail about Scrum on [subsection 2.1.1](#). We will also briefly go through project management in [section 2.2](#).

After that, we will see some background on general Machine Learning ([section 2.3](#)), and later specialize on a machine learning technique called Data Clustering ([subsection 2.3.1](#)).

In [chapter 3](#), we will review some of the works in related areas regarding Agile Team composition, human resource allocation and machine learning for agile project management.

Later in [chapter 4](#) we will dive into the proposal of the solution for the problems mentioned on the previous chapters. The section Methodology ([section 4.2](#)) includes the methods used to collect evidence to support the development of the tool.

2 Background

In this chapter we will describe the background research on Agile Project Management, Team formation and Machine Learning.

2.1 Agile Methods

According to the agile manifesto, Agile methods focus on delivering value. Agile projects often release their product with the most valuable features first and update it over time. When the requirements change (which is almost inevitable) an agile team is ready to change directions to match the current state of the requirements. Agile can, in fact, be described as adaptive ([SHORE, 2009](#)).

Agile methods have 4 core principles:

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan

Before Agile, software development resembled an assembly line. The process was completely documented, the requirements were only defined in one step and there was no place for change. After that, the software was developed, tested and delivered. This process could take months or even years. Although software developed under agile methods can take several months to be built, the customer is always close to the development and any changes can be dealt with almost immediately.

Agile methods brought testing closer to the coding phase, so now, there are regular inspections of the working product. Risks became easier to take care of. Agile made it clear because issues were found early in the project and could be dealt with in no time with low cost. Another thing is: Agile Methodologies use cross-functional teams that are empowered to make decisions about the project they are working on ([DEEMER et al., 2010](#)). This means that there are no definitive, unchanging roles, and the combination of all competences in the team must meet the requirements of the project.

This work will focus on an Agile Methodology known as Scrum.

2.1.1 Scrum

Scrum was introduced by Ken Schwabe and Jeff Sutherland in 1993 (SUTHERLAND et al., 2007). They identified that the methodologies used at the time were not fully capable of dealing with project of great complexity and had difficult change management procedures. After identifying the problem, Schwabe claimed that a new approach was needed to help software development teams deal with changes and complex problems (SCHWABE, 1997). Therefore his work introduces an iterative incremental framework for software development. His methodology was not the first nor the last to introduce self-organizing teams.

2.1.1.1 Scrum Summary

Scrum is an iterative incremental framework for software development. It sets development cycles of no more than a month and are usually called Sprints, and cannot be extended.

At the beginning of each Sprint, usually some stakeholder with the Scrum team, chooses from a list called backlog the features that are going to be developed during that Sprint. Changing the items/features is not recommended.

At the end, the team reviews the developed features with the stakeholders, gathers feedback and starts again with new features (DEEMER et al., 2010). A short summary of the entire development process can be reviewed in figure 1.

2.1.1.2 Scrum Team

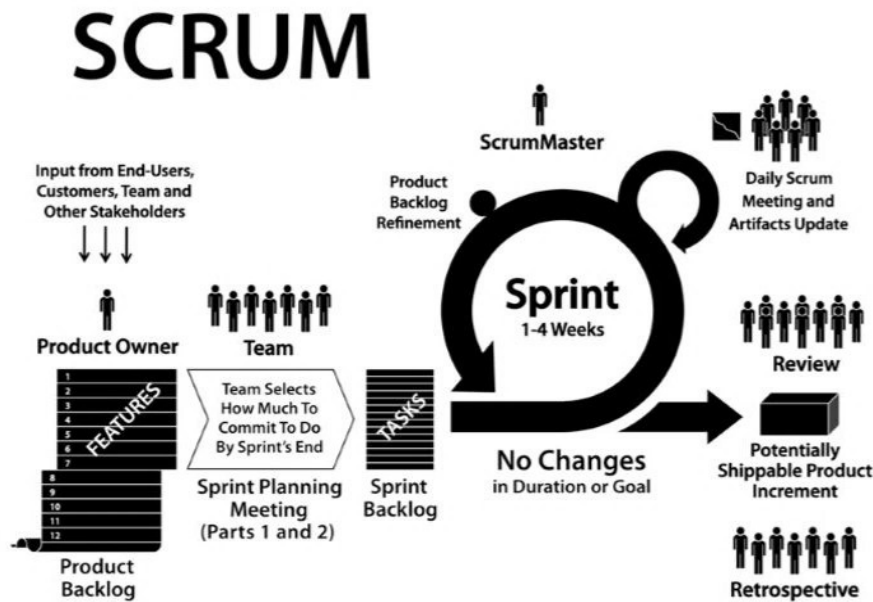
The Scrum team, as mentioned before, is often seen as self organizing. However that does not mean that it is chaotic. According to (MUNDRA; MISRA; DHAWALE, 2013), backed by Schwabe (1997) and Deemer et al. (2010), the main roles are:

- *Product Owner*
- *Scrum Master*
- *Scrum Team*

Each of the roles mentioned above are abstracts at some extent, so they will be broken down into tasks and activities below. The definitions are based on the works of Deemer et al. (2010) and Sutherland et al. (2007).

1. *Product Owner*: This role is one of the most important roles in Scrum, it manages the product backlog and represents everyone with interests in the project. If the team has any doubt about a requirement they should ask the Product Owner.

Figure 1 – Scrum Summary



Source: ([DEEMER et al., 2010](#))

2. *Scrum Master*: The Scrum Master is responsible to make the Scrum process work as intended, to maximize its benefits. He is also in charge of removing impediments with the team or technology that may delay or damage the fluidity of the development process.
3. *Scrum Team*: The Scrum team may vary from company to company but is commonly agreed and also suggested by several different authors that the team should not have more than 10 members including the Scrum Master and Product Owner. The scrum team can be subdivided in smaller roles as seen below:
 - a) *Developers*: The developers are responsible for all the technical work needed to complete and deliver any given requirement. They can be database specialists, User Interface designers, engineers or any other specialization that can help get the work done.
 - b) *Testers*: Testers are the only team members that have an specific task. They are involved in the project from the beginning, working directly with the Product Owner to create acceptance tests. They are also responsible for the integration tests and executing functional tests.

2.2 Software Project Management

The term "project" describes a study or a piece of work with defined activities that has to be finished over a schedule and has an intended purpose. A software project is no different, people have to be managed, work activities tracked and coordinated, progress monitored and so on. All of that, of course, within the scope of a software product (FAIRLAY, 2009).

According to Institute (2001, PMBOK) "Project management is the application of knowledge, skills, tools, and techniques to project activities to meet the project requirements." As stated before, most projects nowadays are using agile, and because of agile being built towards change, project management for agile also is iterative during the project's life cycle.

The PMBOK also divides project management in a 5 Process Group, those are:

1. Initiating: Contains all the processes needed to start a new project or phase.
2. Planning: Contains the processes needed to establish the scope, effort, objectives and actions to accomplish those objectives.
3. Executing: Contains the description of the projects executed to accomplish the project's objectives and specifications.
4. Monitoring and Controlling: Contains all the processes needed to follow, analyze and organize the performance and progress of the project.
5. Closing: Contains all the processes needed to close all the project related activities and processes.

We are not going into further details on each of these categories for the sake of shortness.

Managing a project requires a lot of work, including, but not limited to, addressing the various needs, concerns and expectations of everyone involved, balancing the project's often competing constraints, such as: scope, quality, budget, schedule and so on (INSTITUTE, 2001).

The project team, manager and team members, should be able to balance the demands of the project to deliver a good successful project.

Different from projects in many other areas, software projects are changeable, which means that the Project Manager also has to manage requirements in order to keep things on track.

Managing software teams has been proven to be difficult. Software is a team-oriented, intellect intensive work that requires solving problems creatively. Building teams that will be assigned to a project is the first of many challenges of a Project Manager (FAIRLAY, 2009). Shore et al. (2007), says that any problem regarding software building is, at some point, a people problem.

According to Stellman e Greene (2005) managing a project is all about the team, and trust is one of the most important aspect that a project manager has to have towards his team. Any manager has to rely on his team's expertise to get the job done. However, a manager should not try to micromanage his teams activities, because he/she will certainly get overwhelmed and potentially lead the project to failure.

2.3 Machine Learning

The concept of machine learning comes from the psychological concept of learning. While animals, including humans, learn from experience, machines learn from data. Several mathematical models have been developed to try to mimic biological learning patterns. There is a myriad of such methods available, e.g. Artificial Neural Networks. Machine learning techniques are used to allow machines modifying or adapting their behavior to become more accurate and flexible in their tasks the more they practice (MARSLAND, 2009).

As machines learn through algorithms and data, algorithms can become very complex depending on the problem or even the size of the dataset. There are four types of Machine Learning according to Marsland (2009) and Sammut (2011):

- **Supervised Learning:** Supervised learning refers to any algorithm or computer program that learns through an input and output. Both, input and expected output, must be given for training purposes. After that, the algorithm can be used to make predictions. A classic example of the use of supervised learning is an algorithm that is trained to predict a property value on a given time frame.
- **Unsupervised Learning:** Refers to any machine learning process that learn in the absence of an identifiable output or any type of feedback. Unsupervised Learning is usually used to identify clusters of similar inputs. The algorithm must be able to find similarities itself, within the given inputs.
- **Reinforcement Learning:** Reinforcement learning is the middle ground between Supervised Learning and Unsupervised learning. It uses information saying whether it is right or wrong, but does not say how to correct it. The learner has to try out different techniques and strategies to see what works best for the given purpose.

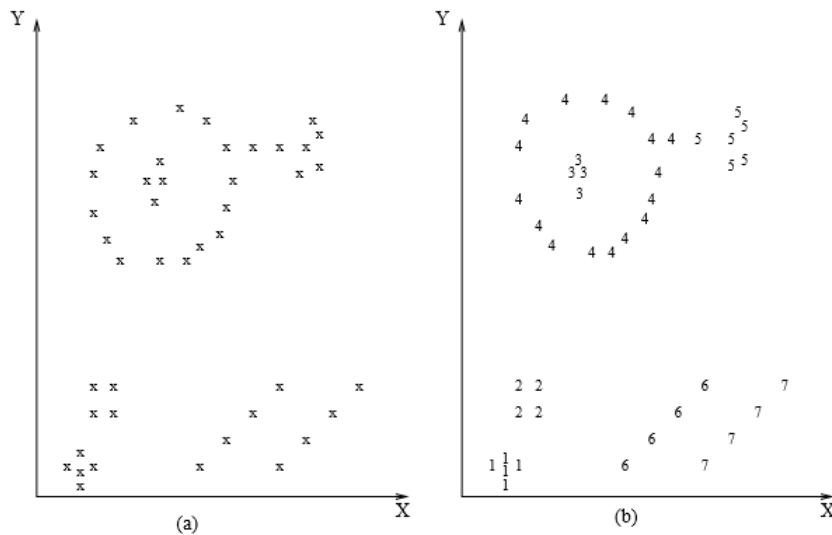
- **Evolutionary Learning:** Evolutionary learning comes from the biological evolution point of view. The possible solutions are like individuals in a population and a function called *fitness function* determines the quality of each solution. Then evolution takes place after the classification of the current population based on its interaction with other individuals inside the population. One example of evolutionary learning are genetic algorithms. Genetic algorithms are heavily inspired by neo-Darwinian evolution theory and can perform most of its biological counterpart jobs such as: reproduction, mutation, recombination and selection.

This work will use an unsupervised learning technique for the problem of Data Clustering.

2.3.1 Data Clustering

Data Clustering aims at organizing data based on similarity patterns between the instances inside the dataset. As seen in [Figure 2](#) the input patterns are represented in (a) and the expected cluster result is shown in (b).

Figure 2 – Clustering Example



Source: ([JAIN; MURTY; FLYNN, 1999](#))

The inputs that are close together are classified as one cluster, and the ones that are far apart are classified into their own cluster with similar counterparts. This classification is often made by calculating the distance from the data point to the center of the cluster. One of the simplest methods of clustering is known as *k*-Means Algorithm, it is based on

a distance measure between the points on the dataset and the mean average to calculate the cluster centroids.

The k -Means algorithm uses a value of k that represents the number of desired clusters. After that it chooses k positions in the map and assigns the cluster centroids μ to those positions. In order to use it, the distance from each datapoint to the cluster centroid must be calculated. Then the datapoint is assigned to the nearest cluster center to the datapoint. There is a step in the algorithm where each cluster centroids should be moved to match the mean of the positions of the datapoints assigned to it. This process must be repeated until the cluster center stop moving (JAIN; MURTY; FLYNN, 1999), (MARSLAND, 2009).

To use it, we must compute the distance of the datapoint to each cluster center and assign it to the cluster.

As said before, the k -Means algorithm is the simplest and most common clustering algorithm. There are several other algorithms with the same purpose that are more suitable for specific tasks that will not be covered here.

We are using clustering because we want to classify our data into similar groups. Since we have the activities of all the agile team members related to a specific project, we can identify their roles based on the data provided by a project management software.

2.3.2 Principal Components Analysis

Principal Component Analysis (PCA) is a multivariate statistical technique that analyses data on a table containing several linear dependent variables that are in general inter-correlated. It extracts all the important information into a new set of possibly less inter-correlated variables. Reducing the original number of columns into a predefined set of variables also known as *principal components* (ABDI; WILLIAMS, 2010). In short, according to Bro e Smilde (2014), PCA aims to identify and select the important information from the data table, compress the data set; thus simplifying its description. For this work, we are using PCA in order to transform a multidimensional matrix of data into a 2D matrix for visualization.

3 Related Work

Several models were built to to solve the automatic allocation of human resources on projects and it is still an issue. The works mentioned below show different approaches to try to solve this very issue.

As an example, [Chi e Chen \(2009\)](#) developed an ontology after having identified more than a hundred properties of individual subjects (e.g. project budget, skill needs, etc). Their work aims at identifying project needs and finding the 'best fit' for team composition. In their work, they filtered to about 35 features or properties and created an ontological representation. They later experimented with their ontological model and showed that it can be used in real world scenarios if provided with a reliable source of knowledge. However their work is not easily applied and requires deep technical knowledge to be automated and used, serving more as a model than a solution.

[Acuña e Juristo \(2004\)](#) analyzed behavioral competencies, focusing on responsibilities and capabilities required by each project. Their model looks into each role and each personal profile available and tries to find the closest match between them. Some of the characteristics of their work is the reliance of management entries of profiles, personal and project roles and constant updates on each individual capability.

Another approach, taken by [Hoda, Noble e Marshall \(2010\)](#) tackles directly the core of the agile development, the self organizing teams. They identified six informal roles that act as facilitators and are present in every agile team. After identifying these roles, they classified them as seen in [Figure 3](#). Their work is limited by its aim, where they focus mostly in interpersonal skills and ignore most technical abilities.

Role	Definition	Played by	Interacts with
Mentor	Guides and supports the team initially, helps them become confident in their use of Agile methods, and encourages continued adherence to Agile practices.	Agile Coach	Team
Co-ordinator	Acts as a representative of the self-organizing Agile team to coordinate communication and change requests from customers.	Developer, Business Analyst	Team, Customers
Translator	Understands and translates between the business language used by customers and the technical terminology used by the team, in an effort to improve communication between the two.	Business Analyst	Team, Customers
Champion	Champions the Agile cause with the senior management within their organization in order to gain support for the self-organizing Agile team.	Agile Coach	Senior Management
Promoter	Promotes Agile with customers and attempts to secure their involvement and collaboration to support the efficient functioning of the self-organizing Agile team.	Agile Coach	Customers
Terminator	Identifies team members threatening the proper functioning and productivity of the self-organizing Agile team and engages senior management support in removing such members from the team.	Agile Coach	Team, Senior Management

Figure 3 – Facilitator Roles in self organizing teams ([HODA; NOBLE; MARSHALL, 2010](#)).

Colomo-Palacios et al. (2012) proposes a system called ReSySTER that is a hybrid recommender system that aims at helping software project managers to better assign resources to projects. It uses fuzzy logic and rough sets to evaluate resources and allocate them in the most suitable project. Despite being very promising, the system requires a technical administrator and its data is collected via empirical data provided by the project manager. Moreover, the system needs input based on the project manager experience. Therefore, its classification is mostly based on empirical data.

One of the most complete works found was '*A team formation model based on knowledge and collaboration*' (WI et al., 2009). Their work evaluates employees knowledge with a technique called 'Know-What, Know-How, Know-Who' to select team members and project leaders. They use quantitative analysis with fuzzy models on the candidates publications to evaluate their knowledge, not only that but they also use their publications as a source for the 'Know-Who' step of the process. This last step evaluates not only their knowledge but also his peers mentioned in the publications, creating something similar to a social network. Their work focuses on teams of scientists. However, it could be used in the assignment of software teams, although the knowledge base would have to be re modeled and only works for the assignment, it does not provide a model for constant evaluation throughout the duration of the project.

Ferreira, Souza e Silva (2010) utilizes genetic algorithms that consider the candidates technical and personal abilities alongside with their preference in regard to an activity. Their model cover almost all the aspects of an optimal team formation, but it does not provide any model or method for continuous evaluation. Their work contributes with an objective function that calculates the total of the candidate's skills plus his preference for an specific task. The objective function is represented below:

$$C = \sum_{h=1}^H \sum_{n=1}^N A_{kn} \beta + \sum_{n=1}^N Pr_n \alpha \quad (3.1)$$

C is the optimal candidate considering that H are the skills required for each activity and h is the skill index. N is the number of candidates and n is its index. A_{kn} is the skill h of the candidate n required. β is the relevance factor of the skill in relation to the activity. Pr_n is the preference of the candidate n for an activity and α is the relevance factor of the preference attribute.

Genetic Algorithms for Project Management by Chang, Christensen e Zhang (2001) defines a series of algorithms that can create software teams with efficiency, although it is a great work and according to the authors one of the best uses of genetic algorithms in project management, it relies in external forms of receiving knowledge. Classification, roles and tasks are not defined by the authors, their tool seeks to be a way of using the project manager's knowledge in tasks and activities required to better assign a software team. As the scope of this work goes, we are trying to define this tasks and

roles specifically so this type of tools can be used in many different projects and contexts.

Allocation and team maintenance is still a problem. A lot of research is being done on the field. All the solutions and researches described above tackle one aspect of the problem. Some of them, as an example [Colomo-Palacios et al. \(2012\)](#), developed a tool using machine learning techniques to solve this problem. However, it is clear that some aspects are being ignored, such as technical skills and continuous verification.

4 Proposed Solution

4.1 Presentation

As seen in Related Works ([chapter 3](#)), most of the tools and techniques used to verify team compositions in agile teams lack accounting some of the technical aspects of the team. When a project manager needs to make a report or needs to identify what his teams are doing, he will also need to understand all the roles played by his team members and check if the roles established are being properly executed.

Most of the work of a Project Manager consists of decision making. The more information available before making a decision, the better. We intend to develop an intelligent tool that will help the project manager.

4.1.1 Design and Implementation

We designed and developed a tool that analyses and classifies the team members into groups of roles. The groups are created using *Clustering* methods with information provided by the team members themselves. With that, we can identify patterns and roles inside the agile team. The default *clustering* algorithm used by the tool is the *k-means*. To develop the tool, we used Java as our main programming language, PostgreSQL as

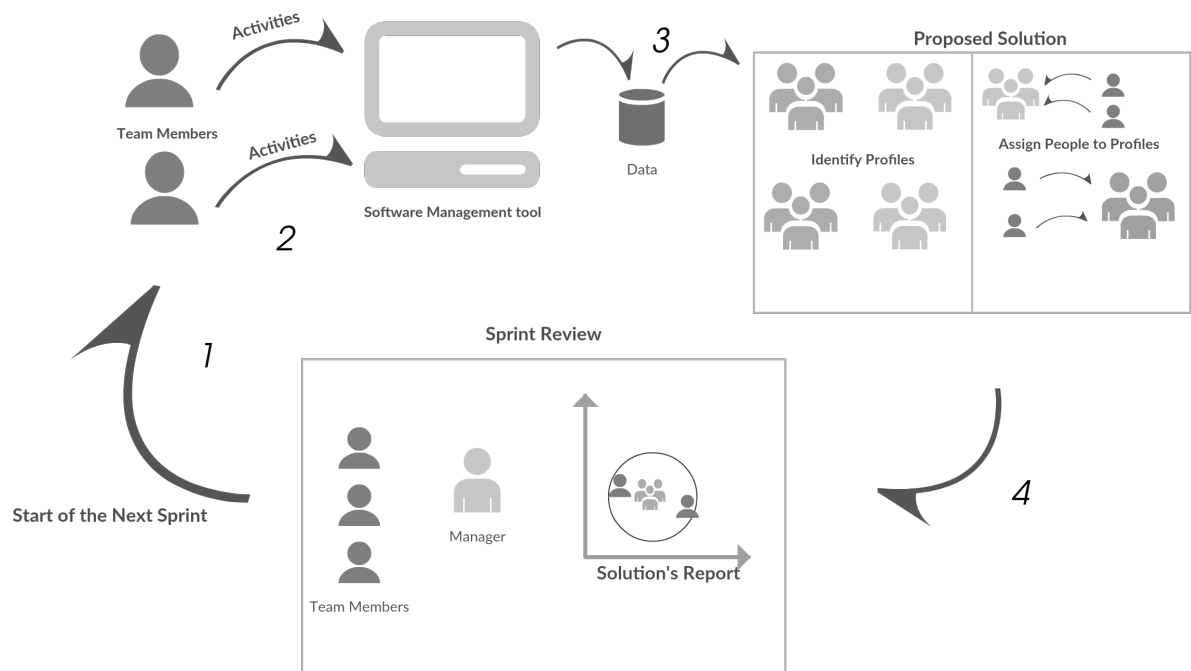


Figure 4 – Diagram showing the proposed solution being used with Scrum

our relational database, the Weka library for machine learning algorithms and Maven to handle dependencies. As of now, the interface was developed using JavaFX and FXML.

The main process for the proper use of the tool is shown on [Figure 4](#).

The cycle begins with a new Sprint (1). The development team uses the software management tool as they are used to (2). The proposed solution takes data from the team's input (3) and applies machine learning algorithms to create profiles based on the team members activities and assign the members to them (4). After that, the manager can see the tool's report, which will deliver information about the team members activities as well as groups of similar activities and members, and use it to support his views during the sprint review.

The tool's report will contain information about the profiles created and their members. The information provided should help the manager to identify problems with more accuracy and also predict some development and problem trends. This way, the project manager would have the necessary information to help his team to achieve better results in the planned time.

4.2 Methodology

First, we tried to find academic journals and books that defined the basic profiles needed on a software development team. Unfortunately, as agile development relies on teams expertise and have no defined roles, we could not find suitable answers for our needs. As a result of that, we could not find, on the literature, the technical skills required for each role.

After that, we decided to create our own sample model for representing individuals inside a software development team. Our model was created using information collected from big tech companies websites. From that information we created feature vectors with the most common activities. For each profile, we selected job offers randomly and from the job requirements we collected the activities and knowledge required (features). Later, after creating the list of features, we decided to use booleans to represent the presence of the skill on each particular job offer. Continuous normalized values could also be considered.

$WDA = [1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0]$

$TPB = [0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1]$

Above we can see the feature vector for an approximate profile of a Web developer from company A (*WDA*), and one vector for a Tester Profile according to company's B website

(*TPB*). To improve readiness, we created a table with the vectors and its corresponding features. The vectors are represented by the columns and the rows are the features that each of the profile needs.

Table 1 – Approximate Web Developer profile collected from 4 different companies websites

Web Developer Profile				
Company/ Features	A	B	C	D
Plan (Design)	1	1	0	0
Code	1	1	1	1
Unit Test	1	1	1	1
Integration Test	1	0	1	0
Unix	1	0	1	0
Prototype	1	1	0	1
Refactor Old Code	0	0	1	0
UI Design	0	1	0	1
Database	0	1	0	1
Test Automation	0	0	0	0
Test in production	0	0	0	0
Debug	0	0	1	0

Considering [Table 1](#), we can for example, compare and sum the different values in each feature, known as *Hamming Distance* ([BOOKSTEIN; KULYUKIN; RAITA, 2002](#)), to calculate the difference between the profiles. In the case of sample profiles A and B, from [Table 1](#), the *Hamming Distance* of the profile A in relation to B is equal to 4.

Other profiles created using the same methodology were Tester and Software Engineer. Those two tables are available on [Appendix A](#) (Tables 4 and 5). It is important to notice that all the profiles created use the same activities. We use that so we can compare them against each other. The used dataset created based on the tables mentioned can be found in [Appendix B](#).

The results of the clustering using the the *k*-Means algorithm with 3 clusters were initially proven inconclusive due to the size of the data set. To obtain an accurate solution, we would need to get more data samples for each of the profiles as well as identify more activities performed by each of them. Although, this experiment was very useful due to the fact that we have experimented with different techniques and algorithms that were, at the end, used on our solution.

After using and trying different techniques with WEKA, we decided to identify the roles and tasks by asking members of several different worldwide tech companies. The analysis of the answers are on [subsection 4.2.2](#).

4.2.1 Collect Evidence

We created a set of questions to assess the need of the proposed tool. Such questions are listed below. They were sent to 15 employees from 3 different worldwide tech companies.

1. What has been your main role, considering the last five years?
2. Regarding your experience in the area of software engineering, what are the most strategic profiles for building a development team? For example: Experienced/Full Coder, Experienced/Full Test Engineer, Experienced Tester, UX Designer, DB Analyst, Project Manager, Technical Leader and any others you might consider relevant.
3. Related to your previous answer, could you list about 5 of the main activities you consider fundamental for each of the profiles?
4. In your opinion, when choosing a developer for a software project (low/medium/high complexity), what is the preferred balance between technical level (knowledge and/or learning capabilities) and interpersonal skills?
5. Generally speaking, in software development teams, what is the highest acceptable knowledge/skill gap among members with similar assignments? For example, when is it acceptable to have an expert engineer performing the same task as a beginner? What could be some effects of such disparity, in your opinion?
6. According to your experience, what are the risks involved (in the short and long term) regarding the build of a software development team with disparities between the member's skills and the wished team profiles? Please consider low and medium levels of disparity.
7. In a software development, how do you see the engagement of project members in tasks that do not directly relate to their profiles/skills/knowledge? For example, what would be the risk of a technical leader being deeply involved in management activities or a test engineer writing production code?

4.2.2 Analyzing Answers

With the answers to the questions shown in section 4.2.1, we can identify if such a tool is needed and how it could be designed. For that, we will analyze each individual answer and compare with all the other answers on the same topic. Only general agreement on any given topic will be considered for this work.

At the start of this writing we received back a few answers.

The answers obtained were from professionals with at least 5 years of experience. Covering a few areas, like research, development, technical leadership and management.

Most of the interviewees seem to agree that a team of software engineers and developers is, in fact, the most acceptable team formation. A software engineer affirms that too many different roles in a software team tend to fail and agrees with 3 other interviewees. Most of them said that different levels of engineers performing all the tasks required for the software product is ideal.

However, a software developer and technical lead, stated that if someone was hired to do one specific task, and ends up doing something different for a long time it could create problems, such as delays, overloading the other with their activities, or even doing something already being done or completed, thus creating a potential risk. Another respondent agrees saying that the variation of activities if performed too often, is never a good thing. It can lead to delays or even compromise the success of the project, other respondent states that this variation may not be a big problem if it only happen sometimes.

The most common roles cited by our respondents were:

- Developer, with main activities like, develop, design, test, code, fix bugs and research.
- Tester, design tests, design and implement automated tests, create test plans.
- Software Engineer, has most of the same activities as a full developer and also gather requirements, review, document and deploy
- UX Designer, implement, research and prototype.

On a side note, it is interesting to notice how all our interviewees mentioned, at some point, mentoring. Which means that even when a development team has a great level of skill disparity, spending time to teach other engineers is not only common but very encouraged.

With all this evidence, it is clear that specific roles are very rare on real world scenarios. This leads to two different conclusions. The first one is that it is hard to define roles in a software team; and software engineers and developers must have the most diverse skill set as possible. The second one is that there is still a need to proper define this roles. As seen on the research done to create the data set, companies still look for specific profiles to fill holes on their teams. Furthermore, the bigger team of generalist developers and software engineers, the harder it gets to identify specific activities and characteristics of each individual.

The solution developed by the authors identifies team members and roles by groups of similar activities and tasks. With this information, the project manager has the authority and data to give the proper feedback for each group of team members.

5 Role Identification Platform

RIP (Role Identification Platform) uses data collected from the Atlassian JIRA software and applies machine learning algorithms to the data obtained. The objective is to identify different roles played and activity patterns inside the software development team.

5.1 Design

In this section, the design steps of the software will be explored. At first, we started developing the software without any established process. However, during the development of RIP we felt the need to establish some practices to organize the work and activities needed to accomplish our goals. After a month of development, we started using TDD alongside with XP. This decision, allowed us to test, implement and prototype easier with very small extra work. Our sprints lasted about a week, where all the progress done was presented and evaluated.

In [5.1.1](#) we will look through some of the user stories that were written to guide the development process. In [5.1.2](#) the software architecture chosen will be presented and the most important design documents will be shown. Later at [5.1.3](#) we will go through a brief overview of how the software management tool was integrated with our Platform.

5.1.1 User Stories

Following our objectives already stated in [1.2](#) we wrote user stories that would later guide the development of the tool. The table [2](#), shows the user stories and their statuses.

All of the stories were broken into very small activities that could not last more than a day's work. Some of them, were not approved at first for not meeting all the requirements and we had to start over.

5.1.2 Architecture and Design

5.1.2.1 Architecture

The architectural model ([Figure 5](#)) of the RIP platform it is a slightly modified version of the common MVC architectural pattern. We have chosen to use the Repository Design Pattern that uses Data Access Objects (DAO) to handle with the database information writing and retrieval.

User Stories			
ID	Description	Priority	Status
US 01	As a Project Manager I want to see in which group each of my team members is a part of.	HIGH	DONE
US 02	As a Team Member, I want to set my own activities according to my work	MEDIUM	DONE
US 03	As an user, I believe that all my data should come from my own JIRA Server instance	HIGH	DONE
US 04	As a project Manager I want to be able to tinker the settings of my clustering algorithm	LOW	DONE
US 05	As an user, I want to access past reports made by me	MEDIUM	DONE
US 06	As a Project Manager, I want to choose the Clustering Algorithm	LOW	DONE
US 07	As a Team Member, I want to create new activities at any time	MEDIUM	DONE

Table 2 – User Stories

We also added external Weka Libraries to our project in order to get stable and well tested machine learning algorithms.

For the View Component, we are using JAVA FX with FXML. We can later extend the FXML and stylize it with Cascading Style Sheets (CSS) in order to suit clients needs without having to redesign the whole application.

5.1.2.2 Domain

After a thorough evaluation of the User Stories, objectives and JIRA (our chosen software management application) architecture, we found a few entities that could be part of our model.

From those we filtered and produced a very high level domain model seen in [Figure 6](#) that demonstrates the relationship between classes.

5.1.3 JIRA integration

After studying several software management tools, we decided to have JIRA as our default application. The main reason is that we already had a server instance license. Other reasons include: its vast library of resources for developers available on their website, well documented software and being commonly used for agile management and facilitated by its plug-ins and built-in functionalities.

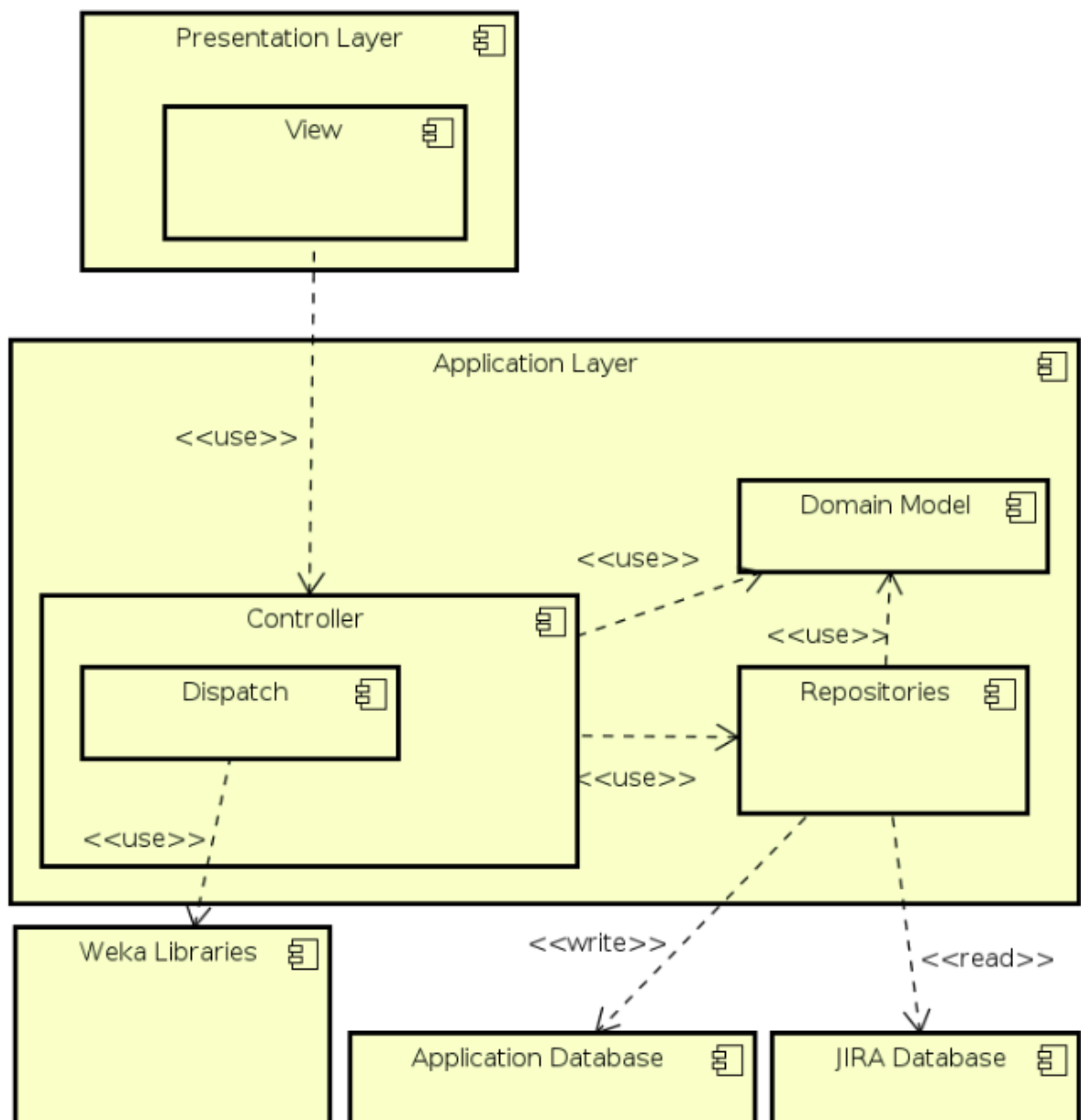
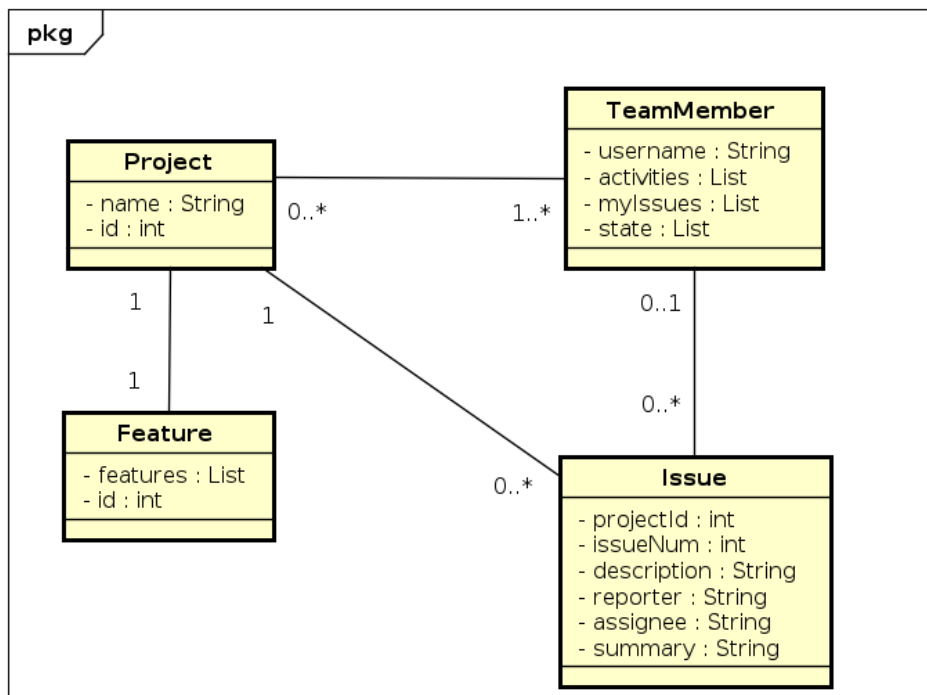


Figure 5 – Component diagram illustrating the architecture model designed for the platform

5.1.3.1 Notation

As a result of the study done on JIRA, we decided to create a notation that could be used to identify the activities that were done to finish each Issue. The process goes as follows:

Before resolving a issue, the team member tagged as the assignee writes on the description what he has to say as normal. After that, he can write the activities completed and time that was spent in each activity. The following notation needs to be used in order to be recognized by RIP.



powered by Astah

Figure 6 – Base domain model used to help visualize the relationship between classes

@activity:TimeSpent

e.g.

Normal description of the solution used to solve issue

@develop:3

@test:1

@design:2

The example shown will be recognized by RIP as *Developed for 3 hours, designed for 2 and tested for 1 hour*. Each activity must start in a new line with '@' (at) symbol in order to be recognized as an activity by RIP, when the name of the activity is finished the user has to use ':' (colon), to be recognized as the end of an activity and finally, the number of hours without spaces just after the colon. It is important to end each activity with a new line, otherwise errors may occur.

Resolving the issue will add the issue to a list that will be later processed.

RIP will run through all the issues that are assigned to each team member and sum all the matching activities. Later this data will be synchronized with all the other team members so they all have orthogonal vectors that can be compared using machine learning algorithms.

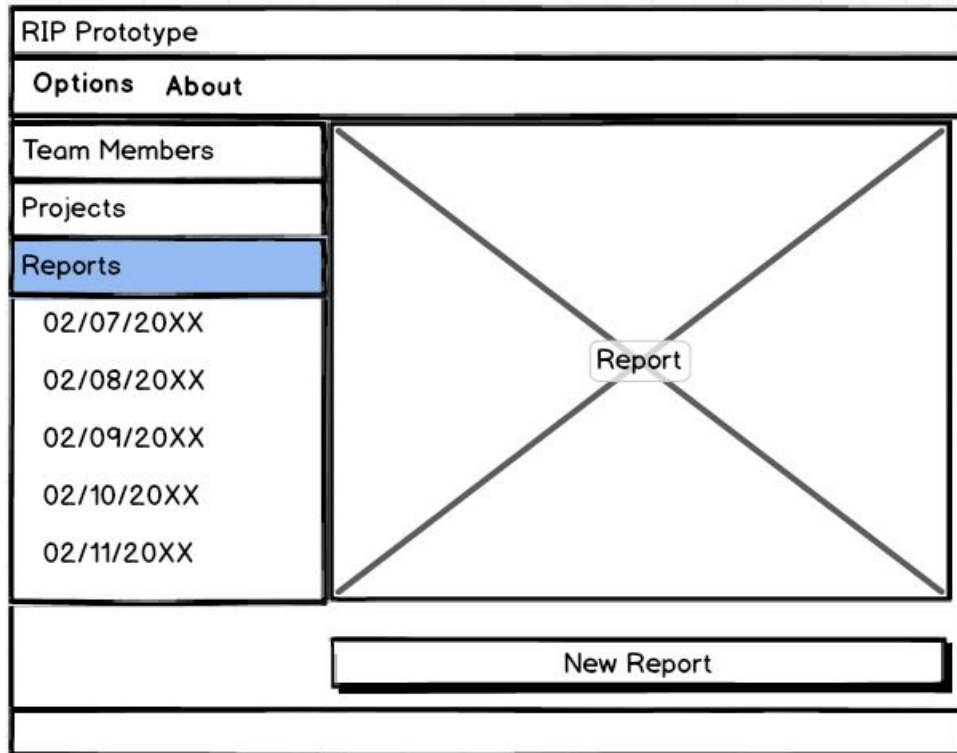


Figure 7 – Final user interface mockup

More detail about the usage of the tool can be found at [section 5.2](#).

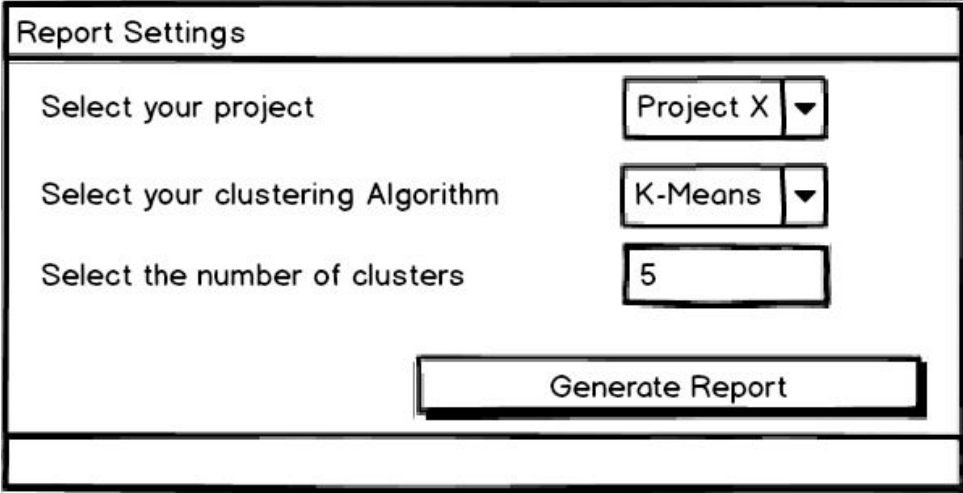
5.1.4 User Experience

After the planning process, the development of the platform was started. For that, planning the User Interface and Experience was a vital part of the process. Using mock-up tools and a very iterative process, the following images represent our prototypes that were later implemented ([Figure 7](#) and [Figure 8](#)). The 'options' menu offers the possibility to load the data from the JIRA Server instance previously installed, offers a "refresh" function to reload the data presented on the interface and a quit option to close the application. You can also find information about the application clicking on "About".

5.2 Usage

5.2.1 Installation

As our application reads data from a third party software, we have to run a few steps to install it and execute it. All the installation steps are well documented and can be seen in a file named *config.file* inside the DevOps folder, Just outside the main src folder. The installation process can be reviewed in ???. We use Maven to build our project, therefore it should not be a problem to build the .jar file and other extensions



The image shows a web interface titled "Report Settings". It has three rows of configuration options. The first row is "Select your project" with a dropdown menu currently showing "Project X". The second row is "Select your clustering Algorithm" with a dropdown menu currently showing "K-Means". The third row is "Select the number of clusters" with a text input field containing the number "5". Below these three rows is a large, rectangular button labeled "Generate Report".

Figure 8 – Final user report settings interface

needed. As an enhancement to the project for future updates, that it is not on our original scope and schedule, a developer may write a script that shall skip the DevOps phase thus creating a fully automatic installation process.

5.2.2 Using JIRA with RIP

After the setup process is done, the development team should use JIRA as it would normally use. Creating issues and assigning them to others. After solving an Issue, it should be marked as "Done" and the description must contain the amount of hours plus the activities needed to solve that issue. The correct format for that input can be seen in (subsubsection 5.1.3.1). A JIRA Issue example can be seen in Figure 9; the issue shown on the mentioned figure, contains all the characteristics needed for RIP to recognize it as a valid entry, which means (a) Its status is DONE. (b) Its description contains the activities following the RIP notation rules. (c) It contains an assignee. What RIP will do is: (a) read all Issues and load all that have the DONE resolution. (b) Create a list of features based on the issue description and (c) assign all read features to the assignee.

Later this data will be processed and reports can be generated based on the information provided by the team members.

Side Note: It is important to convince the development team to adopt a standard naming pattern when creating and adding activities to their assigned issues.

5.2.3 Using RIP

After using JIRA for certain period of time, the Project manager can then use RIP to obtain information about his team. To check if the team data has been successfully loaded, he can click on the side menu and see if all the team members names are listed.

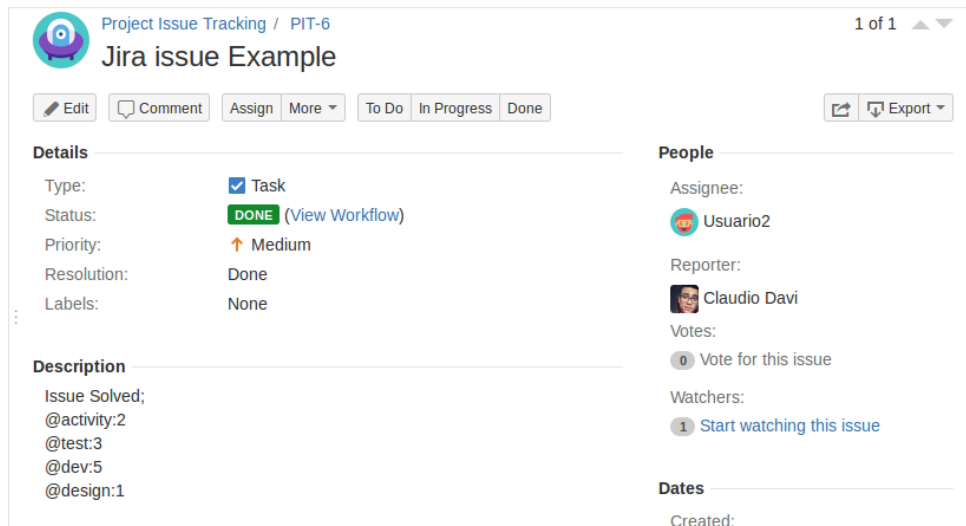


Figure 9 – Solved issue on JIRA interface

The user later can create new reports (after loading JIRA data), clicking on "New Report". You can see the options in Figure 10.

The options window gives the user an advantage to choose which algorithm he wants to use, and the number of clusters he needs to identify.

RIP uses one machine learning algorithm: k -Means already explained in [subsection 2.3.1](#). However, it is possible to extend the code base if more algorithms are necessary.

Another option, "N° of Clusters", comes with a default value of 2. This means that if the project manager wants to identify two different groups of team members with

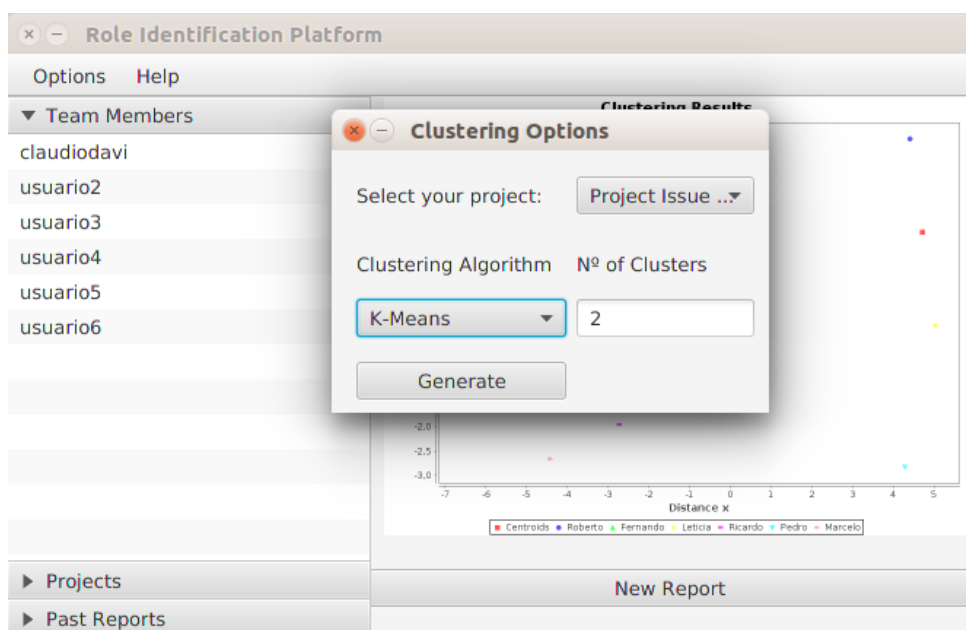


Figure 10 – Clustering Options

related activities, he should keep the default settings on. However, he is free to choose how many groups he wants to identify just by tinkering with this value.

It is important to have in mind that the number of clusters has a major influence on how the data can be interpreted; Too many of them, and the user will get a very sparse set of instances that may or may not reflect the team's reality. One might think that more clusters would mean a more specific group of team members with related activities, which is not true. There is no "rule of thumb" for cluster number, for the use of this tool, we recommend that the user choose a number that he thinks is most fitting for his team, e.g.: There are 3 different sets of activities going on that the same team is involved, therefore, the user can set the number of cluster to 3 in order to classify the members accordingly.

5.2.4 Result Analysis

Figure 11 is a report generated by RIP algorithms after the settings seen previously. For this report, we have used stochastic algorithms to create 6 team members and chosen to classify them into two different clusters using the platform.

Reading the report is quite simple. The red squares are the cluster centroids, they are calculated using the mean average distance between all instances in that cluster. Inside our context, the centroid would be the "average" team member in any particular group of related team members. Looking at the report, on the top right corner we have a centroid and two instances obviously related to it, "Roberto" and "Leticia".

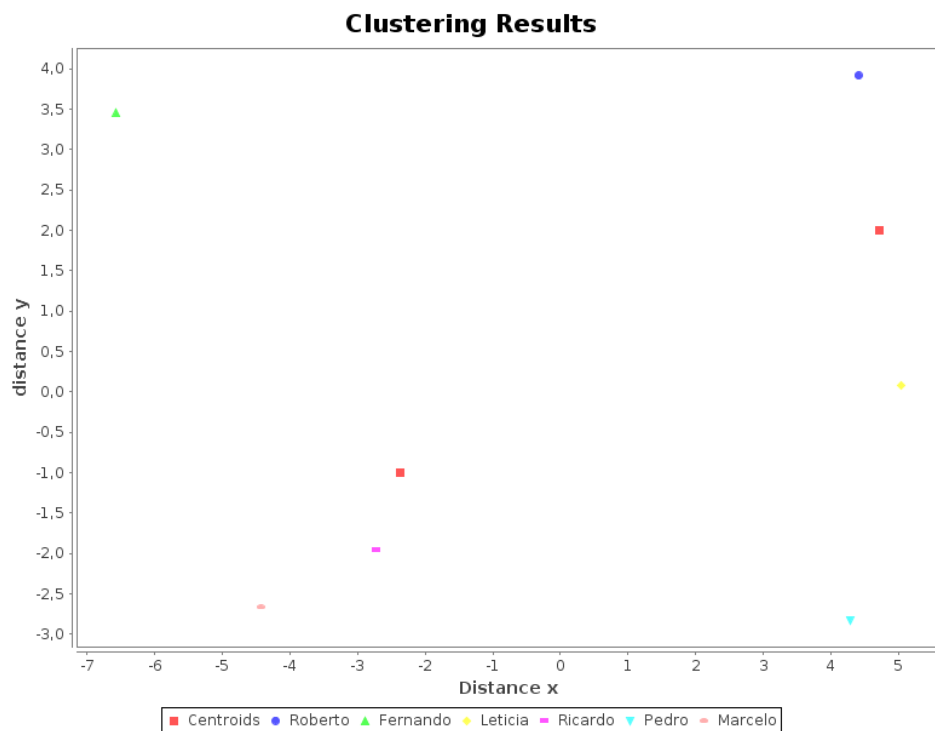


Figure 11 – Report Example with sample data

Looking carefully, we can notice two particular instances that are far from their centroid, "Fernando" and "Pedro". We can see by looking at the position of the centroids, that they belong to the left centroid. However, as they are far away, the user may want to personally check their activities, as they are not matching none of the patterns established by the group. Contextualizing this situation, we may, for example, have a group of developers and testers on the same team. These leads to some questions about the activities that Fernando and Pedro are actually doing, because we can clearly see on the report that they do not belong to any of the existent groups.

As we already stated, the number of clusters centroids chosen is crucial for having a good representation of the reality of the team. We can see in [Figure 12](#) that two clusters does not represent the actual state of the team. However, if the same dataset is applied again with 3 clusters set, we have a much more accurate report, seen in [Figure 13](#).

It is possible that sometimes the user cannot see one of the points in the chart. That happens because the point is probably the only instance of a different cluster, meaning that the centroid is positioned directly above the datapoint. To avoid these inconveniences one should select a different number of clusters to better represent their dataset.

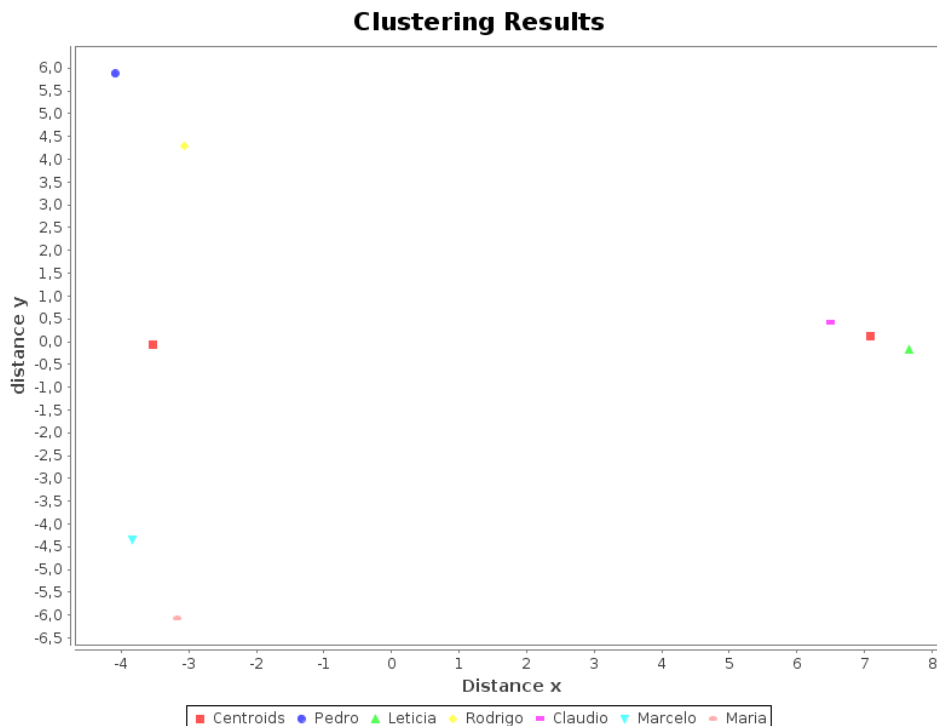


Figure 12 – Report example with 2 clusters

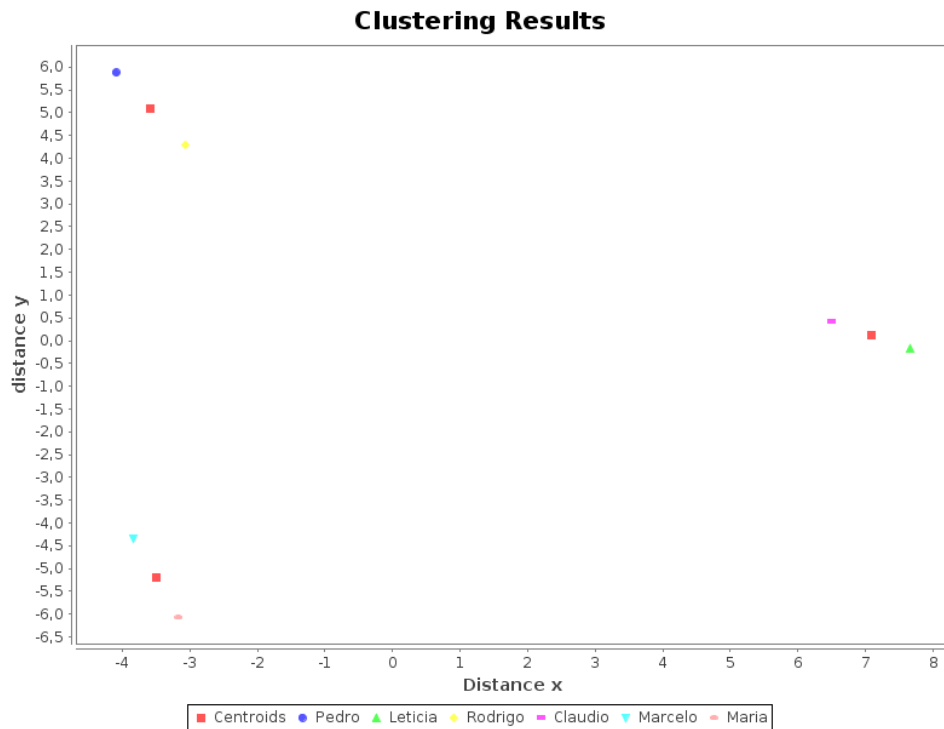


Figure 13 – Report Example with 3 clusters

5.3 Future Works

As seen in previous sections, RIP can be used to identify groups of related team members based on their activities. We achieved our goal in building a functional prototype for the task at hand, although there is still room for improvement.

We believe that RIP can have more machine learning algorithms for the user to choose. The architecture used to develop the tool allows that. This is one of the works that can improve our platform, thus improving the analysis that can be performed by the user.

Another thing that can be done, is to rethink the design. As a prototype, we aimed for simplicity and usability. Our platform was originally built using FXML, which means that it can be extended using CSS. Thus its design can be improved without the use of extra software and frameworks.

Moreover, we intended to build our platform for data visualization and analysis, therefore new visualization techniques can be also a good idea for extending the software. We have created a new visualization model to be implemented, it can be seen in [Figure 14](#).

Some code refactoring could also be done. We have legacy code from our earlier models that can be translated to a more legible and flexible code, some methods can be moved to "helper" classes because they are repeated throughout the code.

We believe that RIP could have a binary option for those whom does not want to

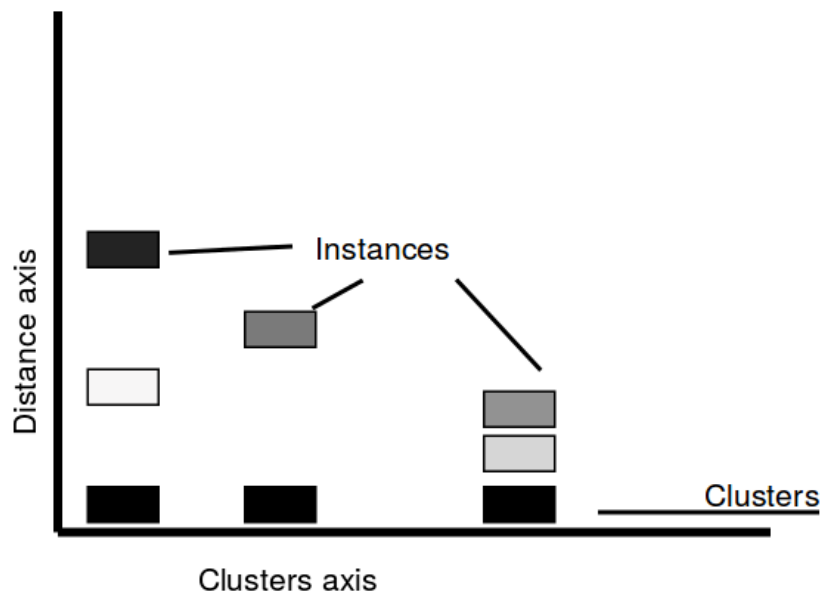


Figure 14 – New visualization model proposal

deal with source code. For that to be possible, we have to extend the platform creating a JIRA Login form from inside our platform. Moreover, a script should also be written to create the database directly on launch, with all the dependencies installed.

Finally, due to the lack of real data, we have not validated our model in a real case scenario. All we have done so far was using empirical data, based on heuristics and experience to check the consistency of all answers obtained. We strongly recommend a carefully and thorough evaluation of RIP into a company or real software development team.

6 Conclusion

We have seen that despite the fact that agile methodologies are well established and widely used, they still have some issues, as in the case of human resources management. In many software development team, teams are assigned to complete a task and are expected to self organize in order to deliver the desired results. This leads to a management problem, where the manager has little to no data on how the team members are individually performing.

After interviewing several professionals, we concluded that it is very hard to define roles inside software development teams. So we now have a manager and development teams that have only a general idea of how everything is working. To solve this issue, we developed a platform that can help the project manager to identify roles and activity patterns inside the development team.

RIP currently is a platform designed to help project managers identifying patterns on their teams. RIP uses a clustering algorithm called *K*-Means to group together members with similar activity patterns. The platform uses JIRA, (a third party software management application), as its main source of data.

After processing the data, RIP will show a report containing information about the team composition. Thus, supplying the project manager with more information to back his decisions up.

As stated on the beginning of this work, our goal was to design and implement a tool that is capable to identify the main roles and team members related to it using machine learning algorithms. With RIP, have all that with the click of a button.

There is still room for improvement. However, we consider the project as a success, since we proved that roles can be identified and patterns in development teams tend to emerge, and with our solution, can now be easily visualized.

Some mistakes were made at the beginning, like not using software processes and reactive planning; those were corrected during the process. We expect that our tool can be useful outside the academia and intend to keep improving it.

Bibliography

- ABDI, H.; WILLIAMS, L. J. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, Wiley Online Library, v. 2, n. 4, p. 433–459, 2010. Cited on page 27.
- ACUÑA, S. T.; JURISTO, N. *Assigning people to roles in software projects*. [S.l.]: Wiley Online Library, 2004. 675–696 p. Cited on page 29.
- BOOKSTEIN, A.; KULYUKIN, V. A.; RAITA, T. Generalized hamming distance. *Information Retrieval*, Springer, v. 5, n. 4, p. 353–375, 2002. Cited on page 35.
- BRO, R.; SMILDE, A. K. Principal component analysis. *Analytical Methods*, Royal Society of Chemistry, v. 6, n. 9, p. 2812–2831, 2014. Cited on page 27.
- CHANG, C. K.; CHRISTENSEN, M. J.; ZHANG, T. *Genetic algorithms for project management*. [S.l.]: Springer, 2001. 107–139 p. Cited on page 30.
- CHI, Y.-L.; CHEN, C.-Y. *Project teaming: Knowledge-intensive design for composing team members*. [S.l.]: Elsevier, 2009. 9479–9487 p. Cited on page 29.
- COLOMO-PALACIOS, R. et al. *ReSySTER: A hybrid recommender system for Scrum team roles based on fuzzy and rough sets*. 2012. 801–816 p. Cited 2 times on pages 30 and 31.
- DEEMER, P. et al. *THE SCRUM PRIMER*. 2010. Cited 3 times on pages 21, 22, and 23.
- FAIRLAY, R. E. D. *Managing And Leading Software Projects*. [S.l.]: IEE Computer Society, 2009. Cited 2 times on pages 24 and 25.
- FERREIRA, F. d. S.; SOUZA, J.; SILVA, J. S. d. V. *Formação de grupos de trabalho com algoritmo genético*. 2010. 1–5 p. Cited on page 30.
- HODA, R.; NOBLE, J.; MARSHALL, S. *Organizing self-organizing teams*. 2010. 285–294 p. Cited 2 times on pages 13 and 29.
- INSTITUTE, P. M. Project management body of knowledge (pmbok® guide). In: . [S.l.]: Project Management Institute, 2001. Cited on page 24.
- JAIN, A. K.; MURTY, M. N.; FLYNN, P. J. *Data Clustering: A Review*. 1999. Cited 2 times on pages 26 and 27.
- MARSLAND, S. *Machine Learning An Algorithmic Perspective*. [S.l.]: Chapman Hall/CRC, Taylor Francis Group, 2009. Cited 2 times on pages 25 and 27.
- MUNDRA, A.; MISRA, S.; DHAWALE, C. A. *Practical Scrum-Scrum team: Way to produce successful and quality software*. 2013. 119–123 p. Cited on page 22.
- SAMMUT, G. I. W. C. *Encyclopedia of Machine Learning*. [S.l.]: Springer, 2011. Cited on page 25.

SCHWABE, K. *Scrum Development Process*. [S.l.]: Springer, 1997. Cited on page [22](#).

SHORE, J. et al. *The art of agile development*. [S.l.]: " O'Reilly Media, Inc.", 2007. Cited on page [25](#).

SHORE, S. W. J. *The Art of Agile Development*. [S.l.]: O'Reilly, 2009. Cited on page [21](#).

STELLMAN, A.; GREENE, J. *Applied software project management*. [S.l.]: " O'Reilly Media, Inc.", 2005. Cited on page [25](#).

SUTHERLAND, J. et al. *The scrum papers: Nuts, bolts, and origins of an agile process*. [S.l.]: Citeseer, 2007. Cited on page [22](#).

WI, H. et al. *A team formation model based on knowledge and collaboration*. [S.l.]: Elsevier, 2009. 9121–9134 p. Cited on page [30](#).

Appendix

APPENDIX A – Profiles Collected

Table 3 – Approximate Web Developer Profile collected from 4 different companies websites

Web Developer Profile				
Company/ Features	A	B	C	D
Plan (Design)	1	1	0	0
Code	1	1	1	1
Unit Test	1	1	1	1
Integration Test	1	0	1	0
Unix	1	0	1	0
Prototype	1	1	0	1
Refactor Old Code	0	0	1	0
UI Design	0	1	0	1
Database	0	1	0	1
Test Automation	0	0	0	0
Test in production	0	0	0	0
Debug	0	0	1	0

Table 4 – Approximate Tester Profile collected from 4 different companies websites

Tester Profile				
Company/ Features	A	B	C	D
Plan (Design)	0	0	0	0
Code	1	1	1	1
Unit Test	1	1	1	0
Integration Test	1	1	1	1
Unix	0	0	1	0
Prototype	0	0	0	0
Refactor Old Code	0	0	0	1
UI Design	0	0	0	0
Database	0	1	0	1
Test Automation	1	1	1	1
Test in production	1	1	0	0
Debug	1	1	1	1

Table 5 – Approximate Software Engineer profiles collected from 3 different tech companies websites

Software Engineer Profile			
Company/ Features	A	B	C
Plan (Design)	1	1	1
Code	1	1	1
Unit Test	1	0	0
Integration Test	0	0	0
Unix	0	0	1
Prototype	0	0	0
Refactor Old Code	0	0	0
UI Design	0	1	1
Database	1	1	1
Test Automation	0	0	0
Test in production	0	0	0
Debug	0	0	0

APPENDIX B – Formatted DataSet

```
% 1. Title: Davi Sample DataSet
%
% 2. Sources: Claudio Souza

@RELATION Work
@ATTRIBUTE plan REAL
@ATTRIBUTE code REAL
@ATTRIBUTE unitTest REAL
@ATTRIBUTE ItegTest REAL
@ATTRIBUTE Unix REAL
@ATTRIBUTE Prot REAL
@ATTRIBUTE Refact REAL
@ATTRIBUTE UIDes REAL
@ATTRIBUTE Datab REAL
@ATTRIBUTE autTest REAL
@ATTRIBUTE ProdTest REAL
@ATTRIBUTE debug REAL
@ATTRIBUTE class {developer, tester, engineer}
@DATA
1,1,1,1,1,1,0,0,0,0,0,0, developer
0,1,1,1,1,0,1,0,0,0,0,1, developer
1,1,1,0,0,1,0,1,1,0,0,0, developer
0,1,1,0,0,1,0,1,1,0,0,0, developer
0,1,1,1,0,0,0,0,0,1,1,1, tester
0,1,1,1,1,0,0,0,0,1,0,1, tester
0,1,1,1,0,0,0,0,1,1,1,1, tester
0,1,0,1,0,0,1,0,1,1,0,1, tester
1,1,1,0,0,0,0,0,1,0,0,0, engineer
1,1,0,0,1,0,0,1,1,0,0,0, engineer
1,1,0,0,0,0,0,0,1,0,0,0, engineer
```


APPENDIX C – Sample Dataset

```
%Author: Claudio Davi Souza
%Role Identification Platform Dataset
```

```
@RELATION features
```

```
@ATTRIBUTE code NUMERIC
@ATTRIBUTE model NUMERIC
@ATTRIBUTE plan NUMERIC
@ATTRIBUTE document NUMERIC
@ATTRIBUTE prototype NUMERIC
@ATTRIBUTE debug NUMERIC
@ATTRIBUTE test NUMERIC
@ATTRIBUTE design NUMERIC
@ATTRIBUTE layout NUMERIC
@ATTRIBUTE reference NUMERIC
@ATTRIBUTE refactor NUMERIC
@ATTRIBUTE report NUMERIC
@ATTRIBUTE decision NUMERIC
@ATTRIBUTE manage NUMERIC
@ATTRIBUTE implement NUMERIC
```

```
@DATA
```

```
0, 0, 0, 0, 0, 5, 1, 4, 3, 5, 0, 0, 0, 0, 0
5, 5, 5, 5, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 1, 4, 2, 2, 4, 0, 0, 0, 0, 0
4, 4, 5, 4, 3, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 3, 4, 5, 2
0, 0, 0, 3, 0, 0, 0, 1, 0, 0, 4, 1, 5, 3, 5
```