BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INFORMATION SYSTEMS
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

IT MONITORING AND MANAGEMENT SYSTEM WITH DCMM

SYSTÉM PRO PODPORU SLEDOVÁNÍ A ŘÍZENÍ IT DLE DCMM

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR Bc. MARTIN VAŠKO
AUTOR PRÁCE

SUPERVISOR RNDr. MAREK RYCHLÝ, Ph.D.
VEDOUCÍ PRÁCE

BRNO 2021

Department of Information Systems (DIFS) Academic year 2020/2021

Master's Thesis Specification

22248

Student:	**Vaško Martin, Bc.**
Programme:	Information Technology
Field of study:	Computer Networks and Communication
Title:	**IT Monitoring and Management System with DCMM**
Category:	Information Systems

Assignment:

1. Study the Digital Capabilities Management Model (DCMM) for ITmanagement of organizations. Also study other approaches to IT management (ITIL or COBIT) and software development (e.g., Scrum,DevOps). Focus on activities in an IT department according to these approaches and on their classification in DCMM.
2. Design an information system that will support IT monitoring and management according to DCMM, from processing of activity monitoring inputs (e.g., calendar events, task records and incidents, etc.) to their evaluation and visualization according to DCMM. Where possible, automate the data analysis process, e.g., by classification.
3. In agreement with the supervisor, choose appropriate procedures and technologies for the implementation of the system. For the data analysis, choose appropriate methods (e.g., a specific classification method), study them in more detail and utilize them in the system.
4. Implement, test and evaluate the proposed system on suitable data.
5. Describe the resulting system and publish it as an open-source.

Recommended literature:

- Zdenek Kvapil, Jonathan Boyd. *DCMM: Digital Capabilities Management Model*. CreateSpace Independent Publishing Platform, 2018. ISBN 978-1723571923
- George Spalding, Gary Case. *ITIL Continual Service Improvement*. The Stationery Office, 2007. ISBN 978-0-11-331049-4.
- COBIT 2019 Framework: Governance and Management Objectives. 2019. [https://www.isaca.org/cobit/pages/]

Requirements for the semestral defence:

- Items 1 to 3.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Rychlý Marek, RNDr., Ph.D.**
Head of Department:	Kolář Dušan, doc. Dr. Ing.
Beginning of work:	November 1, 2020
Submission deadline:	May 19, 2021
Approval date:	May 7, 2021

Abstract

IT development team sees IT management as leaders that keep IT department and IT projects going. Close-mindedness of the company management is seen often as a problem of IT management. However in DCMM terminology, leads of the management should discuss problems between the two parties in even matter. Discussion in such way has positive impact on the IT department employees and the product development. The proposed system allows users (IT managers) to manage and visualise IT department resources, project stage and current IT management of project in DCMM perspective. The main focus is set to extraction of a set of records from various tools that are coupled to stories. These stories forms a sequence of related steps that have positive or negative impact on business. The DCMM metrics were implemented to help IT management decide based on the metric whether to go with new ideas or to stay away from being innovative. While using the implemented system a management is able to choose whether sticking with DCMM helps to be innovative or whether the company should stick with current management model. Based on the current and previous events that happened which created tool shows, there is need to use management analysis to gain new knowledge.

Abstrakt

Developeri v IT tíme vidia IT manažment ako vedúcich ich oddelenia a IT projektov. Úzkoprsosť manažmentu spoločnosti je často videná ako problém v riadení IT. Avšak v DCMM terminológii sa od vedúcich manažmentu vyžaduje rovnocenná diskusia medzi oboma účastníkmi. Pri rovnocennej diskusii to má za pozitívny dopad na pracovníkov IT a riešenia IT projektu, než pri nerovnocennej komunikácií. Navrhovaný systém umožňuje uživateľom (IT manažérom) manažovať a zobrazovať zdroje IT oddelenia, aktuálny stav projektu a aktuálny pohľad na manažment IT v DCMM perspektíve. Hlavné zameranie je kladené na extrakciu záznamov z rôznych nástrojov, ktoré sú spojené do príbehov. Tieto príbehy vytvárajú reťazec navzájom súvisiacich krokov, ktoré majú buď pozitívny alebo negatívny dopad na podnikanie. DCMM metriky boli realizované pre pomoc IT manažmentu aby sa vďaka ním vedel rozhodnúť ísť v smere nových myšlienok alebo sa držať ďaleko od inovácií. Pri použití nástroja, je manažmentu podniku ľahšie určiť, či DCMM pomôže danej spoločnosti byť inovatívna alebo či sa má rozhodnúť zostať v užívaní aktuálneho manažérskeho modelu. Na základe aktuálnych a predošlých udalostí odohraných vo firme, ktoré sa zobrazia je nutné vykonať manažérsku analýzu pre zistenie nových vedomostí.

Keywords

IT management, DCMM, services, software products

Klíčová slova

IT manažment, DCMM, služby, softvérové produkty

Reference

VAŠKO, Martin. *IT Monitoring and Management System with DCMM.* Brno, 2021. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor RNDr. Marek Rychlý, Ph.D.

Rozšířený abstrakt

Úvod

IT manažment spoločnosti je jeden z rozhodovacích faktorov ako IT oddelenie vo firme napreduje a čomu sa venuje. Problémom sú však zastarané manažérske techniky v IT, ktoré vychádzajú zo starých nepružných pravidiel, ktoré pochádzajú zo začiatkov 80. rokov. V týchto rokoch vzniklo ITSM, ktoré reflektovalo IT v oblastiach ako letecký priemysel, bankovníctvo alebo hoteliérstvo. Dnes sa IT zameriava však na vyššie ciele ako distribuované výpočty, klaudové služby a vyvýjanie multiplatformných aplikácií (typicky webové a mobilné aplikácie). V tomto prípade sa musí IT manažér chcieť vzdelávať a napredovať, aby firma rástla. Takéto správanie však vo veľkých firmách vôbec neexistuje. Preto vznikol prístup DCMM, ktorý je založený na motivácií ľudí, vníma IT ako celok a ľudí ako kľúčový faktor k úspechu. Od ITSM sa líši v značnej miere, hlavne agilným prístupom a nízkym administratívnym zaťažením IT developerov.

Riešenie

Táto práca sa zameriava na to, ako vytvoriť informačný systém s polo resp. plno automatizovaným zbieraním údajov z vývojárskych nástrojov. Spomedzi rôznych softvérových riešení boli vybrané Github, Jira, Trello a Google Calendar, ktoré slúžia ako vstup pre informačný systém. Z týchto nástrojov automaticky klasifikuje tieto dáta do DCMM entít. Automatizovaná klasifikácia je vykonaná pomocou modelu strojového učenia nad vlastnou získanou dátovou sadou závad v rôznych IT projektoch dostupných voľne na internete. Klasifikácia prebieha pomocou strojového učenia založeného na spracovávaní prirodzeného jazyka (angl. natural language processing). Medzi DCMM entity patrí aktivita, príbeh alebo komponenta v IT firme. Problém riešenia však nastáva, ak nie je dostatočná presnosť klasifikácie. To môže byť z dôvodu nedostatku dát alebo nekvalitných vzoriek v dátovej sade. Preto je nutné aby bola aspoň 60% pravdepodobnosť, že vstup sa klasifikoval do jednej zo správnych skupín inak klasifikácia vykonáva náhodný odhada a nerozlišuje kontext. Výlsedkom klasifikácie je jeden typ DCMM entity zo štyroch. Avšak DCMM entit existuje 10. Klasifikácia teda nie je úplne automatizovaná, pretože je možné niektoré entity jednoznačne určiť, prípadne jednoznačne neurčiť. Je to v dôsledku chýbajúcich údajov vo vybraných vývojárskych nástrojoch a existujúceho datasetu. Tento fenomén sa rieši v tejto práci tak, že je možné riešiť spôsobom ručného zadania údajov do informačného systému. V prípade, že existuje IT manažérsky nástroj obsahujúci tento typ informácií je možné plne automatizovať klasifikáciu všetkých desiatich DCMM entít zo získanych vstupov. Informačný systém obsahuje volanie API služieb na pozadí a predikciu DCMM entít na základe získaných údajov. Následne tieto údaje sú usporiadané do vzájomných korelácií podľa existujúcich dopredných alebo spätných ukazateľov. Je predpoklad, že IT developeri firmy pri písaní *závad* IT projektu buď píšu tieto ukazatele alebo ich naopak zabudnú a nie je možné jednoznačne určiť, ktorá entita má akého predka priamo. IS však umožnuje tieto väzby doplniť po ukončení vkladania údajov. V prípade chýbajúcich ukazateľov sa vyhľadáva možný predok alebo potomok. V konečnom dôsledku je výstupom jednoduchá vizualizácia DCMM entít pomocou techník popísaných v DCMM.

Výsledky

Pomocou úpravy vstupných dát do klasifikácie sa vytvoril model SDGCClassifier, ktorý funguje na princípe logistickej regriese a učením princípom Stochastickeho poklesu gradientu (SGD). Tento model s pravdepodobnosťou 65% bol schopný správne klasifikovať rôzne vstupy z repozitára `kubernetes` od tých, ktoré sa nachádzali vo vybranom datasete. Nad desiatimi open source repozitármi vybranými v zozname top 100 najpoužívanejších github repozitárov podľa webu `gitstar-ranking` boli vykonané klasifikácie závad, kde pravdepodobnosť správnej klasifikácie bola po získaní 27932 závad bola 54.8%.

Záver

V práci som sa zaoberal implementáciou modelu pre polo/plno automatickú klasifikáciu DCMM entít. Je však zrejmé, že aktuálnym nástrojom voči metodike DCMM chýba správne štítkovanie, ktoré by umožnilo zo všetkých problémov vyextrahovať všetky entity a vytvoriť dokonále prepojenie bez nutnosti dodatočnej interakcie používateľa. To však nebolo možné docieliť kvôli používaným technikám vývoja ako je agilný vývoj, SCRUM alebo DevOps, ktorý nešpecifikuje nutnosť rozdeľovať problémy do príliš špecifických skupín ako DCMM očakáva. Avšak bolo možné zatriediť problém s pravdepodobnosťou 65% do jednej entity z celkovo štyroch DCMM entít. Následne bol implementovaný informačný systém, ktorý sa zaoberal hlavne bezpečnosťou, ľahkým ovládaním pre klasifikáciu a vizualizáciami, ktoré su potrebné pre pochopenie riadenia IT pomocou metodiky DCMM.

IT Monitoring and Management System with DCMM

Declaration

I hereby declare that this Diploma thesis was prepared as an original work by the author under the supervision of Mr. RNDr. Marek Rychý Ph.D. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

. . . .
Martin Vaško
May 19, 2021

Acknowledgements

I would like to thank my supervisor Mr. RNDr. Marek Rychý Ph.D. and Mr. Zdeněk Kvapil. They were both doing the best to supervise my work.

Contents

Chapter 1

Introduction

This thesis is about management of an IT department using Digital Capabilities Management Model (DCMM) approach. Everything that IT department of a company owns and uses every day is also known as a capability. These capabilities are physical and digital where physical belongs to human resources (HR) and employee abilities, and digital capabilities that are expressing what the IT department is capable of in terms of using technologies. Nowadays one of the core problems in big scale companies or in corporate environment is that the company IT department is not reflecting the innovations that are happening on daily basis. The small companies that are ought to learn and innovate their products have better results over time than the companies that are using old technologies. This means that smaller companies are better in managing their digital capabilities than the big companies.

The IT department is one of the core departments where company creates software products or offers software as a service (SaaS). The most important thing is that in DCMM calling software a "service" should be avoided. It is simply because a company offers some unique technology that is innovative and persist to be innovative. This service from Information Technology Service Management (ITSM) perspective is something static and not changing in time because of the Service Level Agreement (SLA). As it is avoided calling software a service similarly it is with "customers". To be able to cooperate between the IT department resources and the customers there should exist equal relationship according to DCMM. It is due to possible improvements and exchange of technology or exchange of ideas with clients or other companies. This is because clients are ought to help, fill support questions and create feedback on the company product or service. The DCMM is strictly against ITSM terminology in terms of the SLA and non equal relationship between buyers of software product and the IT department that product develops.

Despite the use of various technologies or complex tools, the management of company is often not able to understand the reason why to use some technology or tool over another. This is due to not enough technical understanding which is expected by managers. The created monitoring and management system should express to the management how well or how poorly the company IT is working. The common problems in past were that the management did not know whether the IT is working and gaining money without keeping IT software alive or actually performs some actions. The DCMM wants to brings new point of view on how the IT is delivering new capabilities, technologies and innovations and how the IT is not "handcuffed" with past technologies and backwards compatibility.

This thesis brings practical solution for managing the IT department using various of management tools and theirs application programming interface (API) starting from calendar trough project management systems such as Jira and version control system, ending with “thoughts” organizers and communications tools. These tools are used to keep track on what is the company doing, who is assigned to which job and show overall look on the company resources. As the company can fall into unpredictable situations, the DCMM is modeling it as unpredictable activities that have negative impact on the business. However company should mitigate and eradicate issues as soon as possible. As security breaches are the most dangerous events in small to medium businesses, such companies should be aware as much as they can. In case of predictable activities, they are gathered from different sources and serve as an input to the monitoring and management system. Once there is enough input, the story itself can be visualised and the company can decide whether the activities inside story contains the expected amount of resources allocated and whether the story is doable to solve.

To reflect that DCMM is innovative, the proposed system is implemented using various of newest technologies. Mainly the application was designed to be deployed in distributed computing environment where application gathers information from different tools in parallel. These are served as input to DCMM management platform which shows various graphs and feasibility index of the company. There are some limitation based on the possible human mistakes. When parsing from project management system or version control system there are 3 types of issues which refer to particular activity type of DCMM. In case of calendar or “thoughts” organizers there is not specific type of event. However the company can use various calendar types which could determine the DCMM activity type. In case of thoughts organizers tools, the company should separate thoughts to multiple categories.

Proposed solution uses machine learning (ML) model to solve natural language processing (NLP) problem in case of classification of DCMM activities type. This model is filled with relevant data set from the tools to differentiate between DCMM entities and DCMM activity types. However the input should contain exact format so the software is able to perform classification correctly. It is important that input classification should be at least 60% correct in case of solving DCMM classification problem. Otherwise the prediction of the model is just guessing and does not understand gathered description data.

The implementation of the tool is divided to two bigger categories. First one is solving the interaction and scrapping of information from various tools and second category of implementation is solving the visualisation. As it is mentioned first part is considered more technically harder to obtain correct results and to prove that ML model is doing correct prediction of DCMM entities. The visualisations itself are hard in case of showing correct relations between activities or user stories. As the DCMM is quite abstract and big, subset of DCMM entities was selected that are shown by the tool. However the tool was implemented in a way to easily extend the visualisations. It is resolved by storing gathered information in database with dense information.

The results will be separated to evaluation of classification model accuracy, experiments on program correctness and usability experiments. The outcome of the thesis is that the classification data set has it’s disadvantages because it is not balanced. It could not be done due to majority of new features and bug requests and less test, documentation and improvement requests. Using multiple class classification would help the model to help the classification but the data are not correct. The accuracy degrades with number of gathered issues. Even though the classification has it’s problems it still shows the management where they struggle, what are the issues of the current management techniques and whether they should stick with current management. In order to help and raise motivation of employees there is a graph of the most contributing users that helps the product thrive.

The work shows how complex IT management is, and how non trivial it is abstract all of the relationships between entities and predicting multiple labels from text and single label. The implemented solution shows simple visualisation of one concrete DCMM activities. These are easily extensible. The tool shows fast how good is IT department performing IT tasks and how innovative it is. Instead of going trough all of the information manually it is doing fast abstraction of how well the IT is doing and what countermeasures the IT management should consider based on the gained knowledge.

Chapter 2

Approaches to manage the IT department

In project management there exists various approaches to manage the IT department from which selected one in this thesis is DCMM. However without simple comparison why to use DCMM approach instead of others, there has to be performed evaluation whether the company should change current IT management method to another. Namely there exist these management approaches:

- Digital Capabilities Management Model (DCMM).
- IT service management (ITSM).
- Control Objectives for Information and Related Technologies (COBIT) framework.
- Software development (Dev) and IT operations (Ops) DevOps.
- Agile development and Scrum framework.

2.1 DCMM

DCMM first appeared in 2018 where authors were disappointed about management of IT department using previous management methods like ITSM, ITIL or COBIT. The main focus of DCMM is to maximize the entire organisation's digital capabilities, while trying to minimize its own resources for self-administration[11]. How exactly reach the desired aim of DCMM? It lays in being innovative where considering the main factor are human resources(HR)[11]. When IT department employees are happy and have environment in which they can work then being innovative and fulfilling goal is easier than being tied with administrative work and perform no innovative approach[11].

To be able to answer authors question why to be innovative, the answer is rapid technology evolution[11]. A lot of technology were designed and developed recently, where strongest position lays in distributed computing, cloud services, cloud computing, microservice architecture and good ideas that makes our life easier. DCMM thinks that keeping administrative overhead at minimal level saves IT resources.

This approach should be used by CEOs and managers of IT department who like to modernize the way they are managing at the moment[11]. To be able to monitor the management and keeping the goals of IT management the organisation should be able to answer simple question[11]. What is IT doing?

DCMM is inspired mainly by biology and evolution. There exists Digital Agent (DA) that represents biology and evolution. DA is made up of a combination of material (technology, hardware), and immaterial (software, algorithms) elements[11]. The output of DA are digital capabilities. However the DA digital capability can have also negative impact. The DA is going trough life cycles that are non-linear which corresponds with evolution[11]. These DAs helps the IT to solve routine tasks and gain knowledge faster about what IT is doing or what are employees do while they are at work.

When a company wants to be successful the DCMM book mention 5 principles.

1. Sharing.
2. Self-motivation.
3. Uncertainty.
4. Adaptations.
5. Fairness

Sharing means that the IT department may combine resources with other departments, there might be ad-hoc decisions to invest in new technology to explore the potential and possible extension of 1digital capabilities[11]. For self-motivation ITSM used something called as key performance indicators(KPIs), which should evaluate what is the performance of a human entity[2]. DCMM is not agreeing with such point of view and thinks that a person is self-motivated when he or she is doing something that makes sense. Then the person is improving and doing interesting work that fulfills person's needs[11]. Uncertainty is described as unpredictable change that could affect IT in positive or negative way. IT department should be able to adapt on the new situations that happens and be innovative. Fairness describes activity where resources are divided in fair manner[11]. In such case no one in the company means more than someone else even though humans differs. The employees, managers and others that are in company forms one environment which separates to smaller environments of employees. The main focus of DCMM is to keep happy these environments and be equal.

2.1.1 Activity, Component and Story

The main focus in this thesis is on three core components of DCMM which are related to monitoring of IT activity. It is possible to measure performance and innovations with these components. Activity ticket consists of:

- Time
- Type of activity - Normal Status(NS), Extension (EX), Check(CK), Modification(MO), Reduction(RD), Emergency Response(ER), Recovery(RC) and Wrong doing(WD).
- Description what needs to be done.
- Component ID.
- Story ID.
- Assigned employees.

- Status - New, In progress, Waiting for other activity, Put on hold, Finished.
- Result of activity. Summary including simple text with dense information, status and resources spent.

types of activities are visualized on figure 2.1. It is two dimension graph where we can tell about the activity whether it has positive or negative impact on digital capabilities and whether it exhausts or adds to organisation resources.

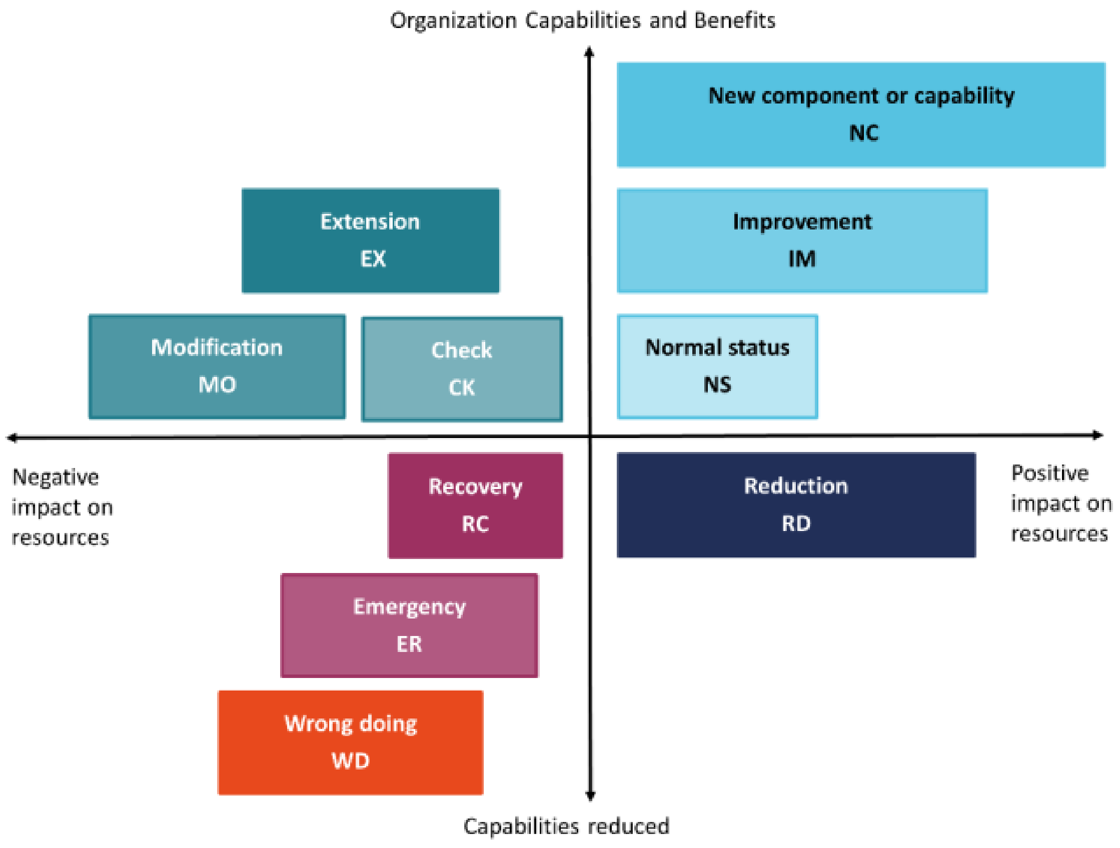


Figure 2.1: Activity types

In this thesis it is focused mainly on 7 of the activity types. At first it is expected that organisation describes what is NS in their organisation. Classification of events from versioning and project tracking software will result in 6 activity types. It can be easily classified what is IM and NC activity and which event is WD. Software is also able to differentiate between EX, MO and CK. However, talking about RD, RC or ER activity is hard because software cannot classify whether event causes reduction. It is possible only if it is explicitly evaluated as reduction on prescribed resource or multiple resources. The same applies for recovery or emergency where must be connection between incident. Such attributes must be inserted in source which these tools does not support.

As next DCMM has record type component. Activity is often related to certain component such as piece of infrastructure, DA, database, data storage and many others[11]. The component has its own description and history of activities and how much resources were spent trough activities on the component. Also it can have relationship to other components[11].

The most value for management is given from stories. Story is chain of activities which are logically interconnected with each other[11]. Start and end of a story cannot be predicted as it appears based on the present situation. For example imagine that software is running and from unknown reason just stops. The story „Software inaccessibility“ starts with first activity called „Examination of software logs“. This story can happen, there is very low possibility but the story appears in moment when the software stops. A story navigator is person that is responsible to decide what activities are going to happen next. The story itself is formed of Story ID, Status, Top level description, story navigator, outcome of related activities, resources, short summary and benefits. The story itself can be split into multiple stories which continues independently and a single story can be part of more stories.

At the end of the story there are three unique information obtained. First is lesson learnt which should consists of gathered knowledge what happened and if it happens again IT management can have guideline how to solve same story again. Secondly there are gathered steps to avoid the story appearance in case if it has negative impact. At last there are created right practices for solving similar situation.

2.2 ITSM

The whole concept ITSM offers is based on providing service for customer[2]. For this purpose ITSM has Service Catalog (SC) where all of the regular processes are described in order to offer single service[2]. All of the SCs provided by single company are also called as Service Portfolio. Provision of such service is non scalable, does not offer innovations and is signed between two parties without any vision of change over time[2]. Let C be a company with IT department. The management of C wants to implement ITSM for whole IT department. What must IT department do in order to offer single service S?

1. Create precise SC that involves everything that the IT is offering to client.
2. Set measurement formula to evaluate Quality of Service (QoS).
3. Set continuous monitoring of offered service.
4. Create report template that are filled by unsatisfied customers.

The easiest are first and last steps, however measuring QoS and perform continuous monitoring of S causes that IT department is always processing trough reports of customers, measuring QoS and monitoring over and over single service that could be faulty once a year. This scenario shows single service S. Imagine company C_c that is starting its business and offers services. C_c company hires 4 new employees to the IT department. Is it worth to implement ITSM? Definitely not, the amount of resources spent to provide single service for plenty of customers is undo able and sooner or later the IT department needs expansion.

Let assume someone launched an attack A on service S. C has to follow service statements to mitigate any losses and keep providing service S. This process contains concrete requirements that can be found in table 2.1[2].

It is clear that single incident must be held by at least 3 people from which two are technicians and third is solving the issue. However does small company C_c have database of knowledge? This database is created as result of gained experience from past incidents. This is impossible to follow by small company. Does it mean that small companies just cannot follow ITSM? Then why we separate between small and big company when both can reach nowadays the same result by using for example cloud services to run their products?

Record of incident	Resolution of incident using database of knowledge
Actualization of incident	Inform responsible staff and customer/s
Communicate with customer/s	Formally close incident
Personal technician	Head technician

Table 2.1: Requirements for reporting single incident

These questions should fulfil why the ITSM should not be the only standard for managing the IT and there is ought to be some standard for smaller companies. That is why DCMM is the way to approach the rest of organisations while at the same time offers new point of view for big companies[11].

2.2.1 Comparison with DCMM

In comparison with DCMM, IT is not only delivering value for customers in form of IT services[11]. The main idea is to maximize C or C_c digital capabilities, while trying to minimise resources spent on self-administration[11].

The IT is living structure that could be represented as set of HR, capabilities of employees and leadership with combination of digital capabilities[11]. These digital capabilities includes technology used, innovation performed etc. The uneven relationship between "customer" and IT department should work in a collaborative way instead. Also the DCMM points out that ITSM was created to manage IT department of the big businesses such as banking or finance sectors or airlines[11]. The author of DCMM assumes that while ITSM was forming the management of such companies was very strict where they put resources and also careful about terminology used when SC was created[11].

In case of simple example with companies C and C_c, the DCMM does not strictly line up what should IT department do in order to provide service. The mentality of DCMM is "how to provide service with minimal administrative overhead?"[11]. The solution lays in cooperation, maximization of resources and untying the IT department hands by filling number of administrative overhead. Also usage of new technologies and new approaches to create a software the IT department can be self sufficient and can reduce the amount of administrative to the minimum[11]. However it is expected that communication between IT department and management must be involved on even level and not employee to employer level.

2.3 COBIT

According to definition COBIT framework is aiming on whole enterprise IT[7]. Already it is pointed out that small to medium companies cannot implement it. Does it mean that small to medium companies should not have IT management? It is clear that management of IT must be done in every organisation, nevertheless the organisation scale[7, 11, 8]. However in smaller companies there is one manager while in bigger companies there are board of managers where some of them are specialists to IT and some of them are not. The main goal should be the know how of reaching the harmony when decisions are going to be made. There are two things that are discussed in COBIT framework:

1. Governance.
2. Management of information and technology.

Chairperson and board of directors take care of Governance inside organisation[7]. The CEO and CTO takes care of management which involves:

- Management of planning.
- Management of building and running.
- Management of monitoring activities.

To be able to understand the COBIT framework in general there is simple explanation on figure 2.2.

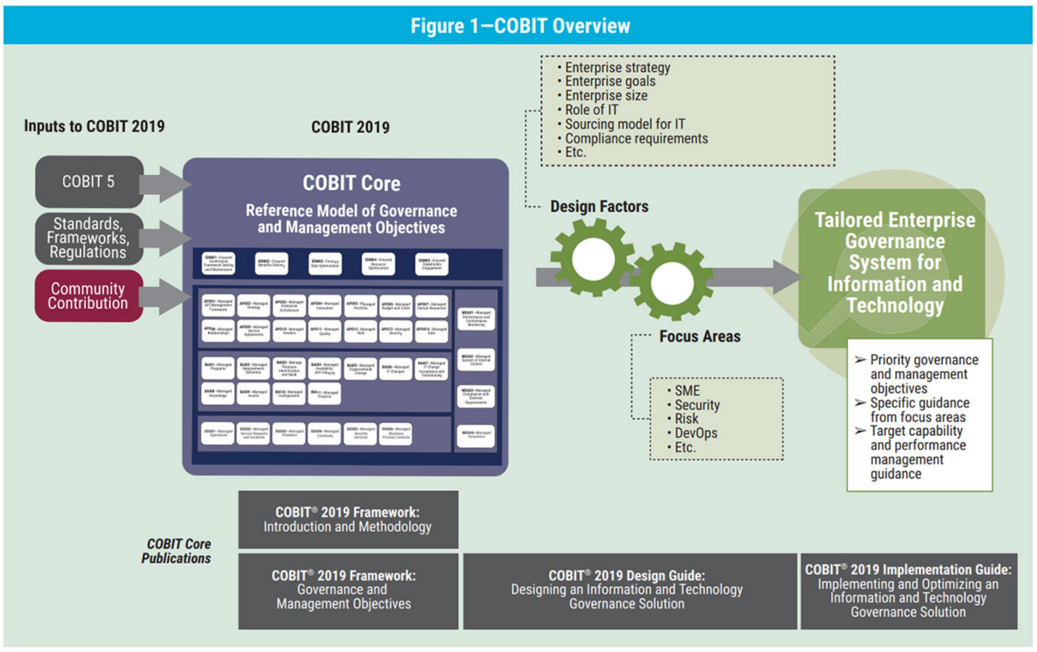


Figure 1—COBIT Overview
Inputs to COBIT 2019
COBIT 2019
COBIT 5
Standards, Frameworks, Regulations
Community Contribution
COBIT Core
Reference Model of Governance and Management Objectives
• Enterprise strategy
• Enterprise goals
• Enterprise size
• Role of IT
• Sourcing model for IT
• Compliance requirements
• Etc.
Design Factors
Focus Areas
• SME
• Security
• Risk
• DevOps
• Etc.
Tailored Enterprise Governance System for Information and Technology
➢ Priority governance and management objectives
➢ Specific guidance from focus areas
➢ Target capability and performance management guidance
COBIT Core Publications
COBIT® 2019 Framework: Introduction and Methodology
COBIT® 2019 Framework: Governance and Management Objectives
COBIT® 2019 Design Guide: Designing an Information and Technology Governance Solution
COBIT® 2019 Implementation Guide: Implementing and Optimizing an Information and Technology Governance Solution

Figure 2.2: COBIT framework. Source: COBIT framework[7]

As it can be seen there are inputs that are mostly standards and regulations that are processed in COBIT core from which is created tailored enterprise governance system. It is a set of specific standards, priorities and management objectives used by big corporate organisation. It is created in order to keep track with organisation objectives. There is lack of interaction with developers and non-managers. It has strict division between managing the organisation and managing the software product.

All of these are tight with direction which is set by the governance body to achieve enterprise objectives. The management and governance has its objectives that are tight with single process[7]. The management processes are typically accountable in difference of governance processes.

The process describes organized set of practices and activities to achieve certain objectives and produce set of output[7].

2.3.1 Comparison with DCMM

Simply, the COBIT framework is typically implemented in big corporate environment where everything should be done in timeline order and objectives are achieved by finishing set of processes. There does not exists much pressure from competitors to use new technology unless management is innovative and wants to be the best in the market. The COBIT framework is not suitable to be used with combination of DCMM because of high administrative overhead and management control over whole company. The DCMM point of view is focused on the cooperation and achieving the best results for the customers while staying unite[11].

There are also similarities between DCMM and COBIT. One of them is that goals form a cascade. The single goal could be reached by achieving smaller goals that forms one objective[11, 7]. In the DCMM mentality it is story based mechanism where single story can be predecessor or successor for old or new story. A story achieves single goal in COBIT meaning.

In both ITSM and COBIT framework the main advantage is that management also evaluates IT decisions based on possible gain or loss and evaluating risks[**?**, 7]. This is also called in ITSM terminology as change management. The management is considering whether it is worth to take project[**?**]. In IT security field it is also considered worth taking risk assessment[18]. This asses developed software against threats and vulnerabilities that have impact on produced solution[18]. Based on the initial budged the IT must evaluate how much time they have to accomplish a task. This point of view is DCMM criticising because when innovation should be made it has many positive effects that we cannot in current time evaluate[11]. From these positive effects IT department in future could considerably benefit. However considering risks in general is always worth. By using risks it is possible to argue whether it is good to take the risk or there is not even possibility to finish the project based on the budged.

2.4 DevOps

The DevOps is closely related to agile development. It follows the agile development which is iterative development including continuous feedback[14]. DevOps by itself does not involve managing revenues and IT in general but managing the IT development team. In ITSM terminology the continuous feedback is measuring the QoS. However in case of IT development in DevOps perspective the management of product consists of managing API, development and operations[14]. There is lack of IT management from resources trough HR but the capabilities of IT are utilized well. When organisation follows DevOps, the workflow of IT is focusing on continuous development to offer the best software services[14]. There are some of the common benefits including:

1. Software updates are made in small regular batches, allowing businesses to adapt quickly to new information.
2. Mistakes or false assumptions are detected much earlier in the software life cycle, where they are much less costly to correct. Also no extra administrative is required since companies using DevOps works with project tracking software e.g Jira.
3. The detection of problems are quickly found and early eradicated. Teams have much faster feedback and organizational knowledge improves. This enhances overall digital capability.
4. By working in parallel across development and testing, agile teams can increase the velocity of application development and delivery.

The main focus is pointed to development and operations, rather than wait for approvals and sign-offs with management of organisation across multiple stages[14]. Even though DevOps does not involve IT management, it manages the product and encourage development team to be able to deliver the best IT services or IT products[14]. Agile methods supports the notion that fully autonomous teams should be able to make their own decisions[14]. It concludes that management of HR and management of IT department is missing dialogue in comparison to DCMM terminology. As the teams are fully autonomous it is related with DCMM meaning of IT department[11, 14].

In comparison with ITSM where systems were mostly monolithic and not regularly updated just to keep the service alive, DevOps wants to operate on single independent units[14]. The reason is that dependent application units often depend on one team to finish it. It increases overall development time of application because another team cannot progress forward[14]. This helps the fully autonomous team to be delivering at faster delivery since the workload is divided between all the resources and capabilities are shared. The main reason to divide workload is to create `Microservice design`[14]. This design offers possible extension of provided service to be operated on cloud or in distributed computing environment and can be distributed to users reliably using load balancing to ensure the user satisfaction[14, 15]. In essence, each microservice focuses on a specific element of business functionality. Microservices can provide business a number of significant benefits:

- Independent coding — Teams have more freedom to develop in different programming languages, each optimized for different processing tasks. Microservices can free organizations from being locked into a single technology stack[14, 15].

- Fault tolerance — When a microservice fails, it is unlikely that the entire system will fail. If the recommendation service fails in a web application, shopping cart and payment processing services can continue to function[14, 15].

- Increased agility — Microservices designs can better support continuous delivery. Since systems are built and deployed independently, they can potentially be tested and delivered faster[14, 15].

- Scaling - The general problem of monolithic software is that it is not scalable and the execution is often coupled with high amount of allocated resources. Often only some parts of application needs higher amount of resources than another e.g Amazon web application does not need high amount of allocated resources instead of running payments[14, 15].

Generally DevOps focuses on building APIs as organisation product. The business behind executing ideas quickly is building APIs that offers new business opportunities[14]. The DevOps ideas using such approach could be better explained on figure 2.3.

The development is divided into API owners that are putting ideas what should API contain and developers that provides these ideas. The actual API is divided into data explorer that are mostly used by users and managers and API creators that use it for future development purposes. The main reason why to create programming interfaces is that the consuming application can be system on chip (SoC e.g Internet of Things or IoT), virtualized application or web service for users. The API communicates with backed such as database servers or authentication servers whether the resources are accessible for regular user or authorized user. These days almost everyone owns mobile phone or multiple IoT devices in their smart homes. When creating APIs it is worth to support these devices because

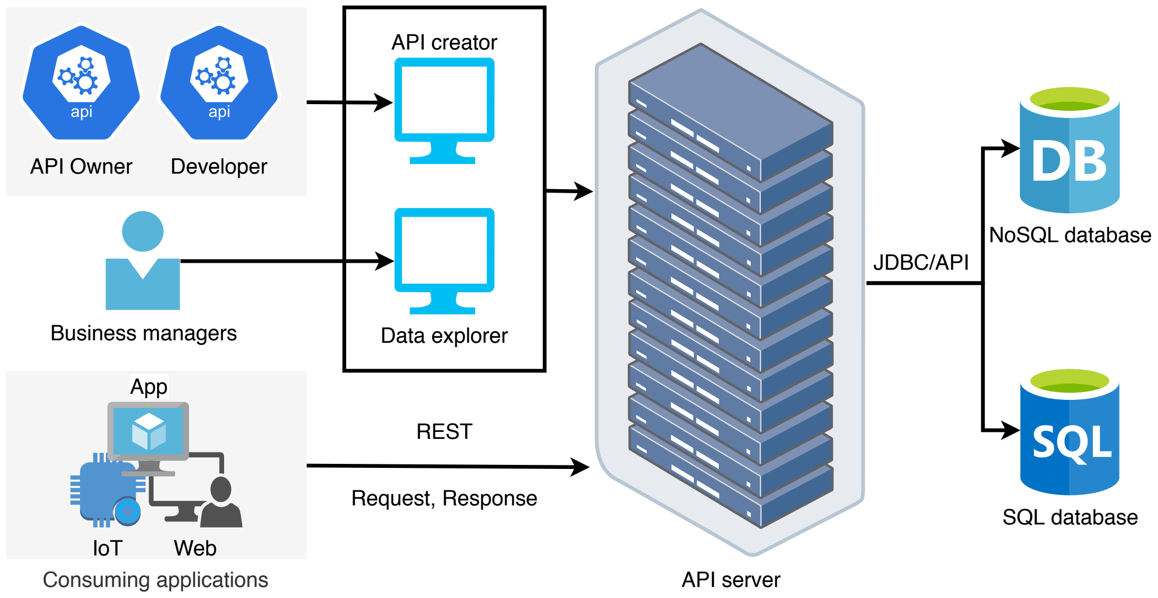


Figure 2.3: DevOps API building and deployment

it allows to reach wide community of users. This phenomenon in DCMM terminology would mean that DevOps could not be confronted with task of being innovative since most of the technologies and capabilities are new. However DCMM is pointing out that being innovative is not only using new technologies and smart application but it is also about research and development (RnD) which could lead to extending existing IT capabilities[11].

Nowadays companies want to deliver exceptional customer satisfaction of their offered services. To deliver such user satisfaction the companies need to create road map of newly offered product and create API that ensure all of the expected goals[14]. Mainly the organisations should focus on measuring the performance of offered APIs because of potential growth of users[14]. Without effective API management the company will struggle to deploy, control and measure API and loss the competitive battle against other companies[14].

2.4.1 DevOps metrics

Since DevOps is not a formal framework, organizations have little guidance in determining what metrics should be used. This can be problematic and lead to a number of sub optimal practices:

- Efficiency status-quo — The IT team falls back to metrics traditionally used to demonstrate technical proficiency in meeting stability and resilience goals. DevOps metrics should also demonstrate how new capabilities are influencing the business.
- Outputs over outcomes — The organizations gravitate to metrics that are commonly used in assessing team-level productivity[14]. These can include output-based metrics like number of contribution to software. Metrics in this class can be counterproductive unless balanced with outcome-centric indicators[14]. The metrics could be compared to desired quality levels to prevent counter production.
- Low-hanging fruit — The organizations select metrics that are easily obtained but not necessarily useful. Since DevOps success is predicated on cultural change, businesses

must also measure namely how the adoption of DevOps behaviors and values at an organizational level is impacting the business.

2.4.2 Case study

According to article of the engineer inside a organisation had responsibility in maintaining large checkout system [8]. Later there were discovered that two monolithic parts of software contained more than 6000 and second more than 4000 lines of code. The problem was that except of maintaining monolithic software the code also did not follow best code practices etc. The main task was to persuade the management, that refactoring is the way of helping the IT department. From culture perspective not everyone embraced the big change. In organisation matter not all of the managers agreed to do refactoring[8].

As it can be seen the main problem is that innovative and proactive approach of IT department is often hard to explain to management. By showing the exact pros and cons and enforcing culture where innovative and proactive approach is necessary, the refactoring solution is the only way out. Using combination with DevOps the refactoring is going to have prescribed properties to maintain it as multiple independent units to split the workload.

2.4.3 Comparison with DCMM

To support the businesses, DevOps offers to innovate where development teams must continuously deliver software services at an increased velocity. The reason why DCMM exists is to manage IT in way of being innovative and company's HR plays important part of being concurrent to others[11]. The innovations and continuous improvement are the key values of IT department[11, 14].

From the culture perspective of IT department DevOps choose trust as one of the most important role. When company does not trust what IT department is doing there is no way that the organisation can be successful. In the development perspective the trust is that production performance information help in a software refactoring and reducing technical dept even though the operation of refactoring has negative impact on IT product from expense perspective[14]. In the operations meaning DevOps put trust in new application design patterns which will help the business scale.

One of the main reasons why DCMM is different from DevOps is that in DevOps vocabulary they still divide between competitor and companion. It is expected that companies can work together to offer single service from which one is prerequisite for the others and are coupled together as the two organisations are co-working[11]. Also DevOps perceive digital transformation as allowing created application to be available on different digital platforms e.g Mobile, IoT etc. The organisation build at the last stage robust digital ecosystem by expanding their partnership to other companies.

2.5 Agile development and Srum framework

The agile manifesto appears as first after ITSM. The main reason why Agile manifesto came up was that IT department was always administratively overwhelmed[4]. Here is a part of quoted statement from agile manifesto:[4]

> Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

It can be easily figured out that the satisfaction of customer lays in frequent updates where user gets new functionalities that could help the user and product. The ITSM on the other hand wants to keep service running no matter what. Once it was designed it cannot be changed and that kind of vision is wrong. The change of requirements comes over time and the software must be updated or changed in positive way. Probably it does not bring new revenues for organisation but it can involve user satisfaction which means much more for users. The application will always attracts new users when it is stable, accessible at any time and offers substantial functionality for them.

On the other hand the agile development see an enemy in competition[4]. That does not always need to be correct vision of others. Sometimes sharing wisdom, resources or digital capabilities between organisation could be beneficial to find even better solution than competitors[11]. However the agile development is paying the attention to have correct design and best architectures of software that fulfills user requirements[4]. DCMM shows that IT department should be managed also in an unpredictable way because of unpredictable requirements[11]. The organisations should be visionary in order to create service or product that no one think about could exists. By achieving that the company can be first in the market and innovative because previous generation did not think about such application.

Scrum is a framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value [16]. To be able to understand scrum framework there has to be explained scrum glossary. We have three aspects in development cycle of software.

1. Scrum artefacts.
2. Scrum events.
3. Scrum team.

The scrum artefacts are divided into `Product backlog`, `Sprint backlog` and `Increment` [16]. The increment is also known as single iteration in iterative development life cycle. Sprint backlog is the set of product backlog items selected for the Sprint, plus a plan for delivering the product increment including realization [16]. An ordered list of everything that is known to be needed in the product development is known as product backlog. The product backlog is never complete as the requirements may change or new requirements can appear. The mentality between product backlog and DCMM activity is tightly coupled. The intention behind scrum artefacts is that the product realization consists of multiple iteration of development. The DCMM is managing the project in no specified way just it has to be innovative and usage of new technology is a way forward. This is why DCMM could combine with some other framework to reach desired quality.

The scrum events are divided into `Sprint`, `Sprint planning`, `Daily scrum`, `Sprint review` and `Sprint retrospective` [16]. The sprint is a time-box of one month or less during which is delivered single iteration of product [16]. During sprint solution the changes cannot be made. Scope of the sprint could be clarified and changes between the `Product owner` and `Development Team` when knowledge is gathered.

The sprint is planned by the `Scrum master` when scrum team has met. On that meeting there should be clarified what is going to be delivered in next iteration of product which results to `Sprint` [16]. Also there is negotiated how to deliver the iteration to the end user. During sprint solution there is `Daily scrum` meeting that should take 15 minutes where development team synchronize activities and create a plan for a day. This meeting takes place every day during the sprint solution. Based on the scrum guideline this should optimize team collaboration [16]. To reduce administrative load the daily scrum takes place at the same time and place each day. For the organisation it is important to keep scrum team happy. The DCMM view is similar in a way that all of the developers are like "family". This involves cooperation helping and being support for others. While working in such environment the scrum team is happy so the organisation is able to deliver quality software.

After sprint is finished `sprint review` is held [16]. This review should inspect the delivered iteration of software product and adapt the `product backlog` when it is needed. There is whole scrum team involved. The product owner explains what items have been finished and what is still ongoing. Yet it could be imagined that product owner is partly a manager of product. This means that this product owner influences the management of the IT project. However in scrum framework there is not strict guideline how to behave or how to manage the team. It is mentioned that team also review budget timeline and marketplace to be sure that the product is still not developed or what could they do better than competitors. Between next after sprint review and before next sprint planning there takes place the `sprint retrospective`. This meeting should involve scrum team without product owner and discuss between development team what could be improved in the next sprint.

In case of the Scrum team it is divided in three different responsibilities [16]. Namely there is `Product owner`, the `Development team` and `Scrum master`. When organisation have multiple software products there are multiple scrum teams that are self-organizing and can work with other scrum teams. The product owner is responsible for maximizing the value of product and expressing what is needed to implement based on the society demand. These demands are clearly expressed in `product backlog`. When priorities of software changes the product owner is also responsible of ordering the items in product backlog. When there are uncertainties in product backlog the product owner is explaining the development team what he meant. The scrum master is a servant-leader for the scrum team. The scrum master helps those outside the scrum team understand which of their interactions with the others have positive or negative impact. The DCMM theory points out that the interaction is important while co-working on the software project.

2.5.1 Comparison with DCMM

There are similarities between agile manifesto and scrum framework. Both are being agile, delivering product in iterations and applies changes during project solution[4, 16]. In case of scrum the deadlines are tight while manifesto does not involve management techniques[4]. The DCMM does not include how to solve project[11]. On the other hand it just wants to point out that the employees within organisation should be happy and feeling good inside provided environment to maximize their capabilities[11]. Also by maximization of capabilities DCMM understands interaction between humans. The product is often developed faster and better while changing the wisdom between humans or organisations[16, 11]. Also DCMM points out importance of being prepared for unpredictable events. Even though the company does not know about any ongoing attack it is preparing defenses to protect either physical or intellectual properties[11].

The IT department is able to deliver solid product with good management while using DCMM in combination with agile development or scrum. DCMM could be combined as well with DevOps theory of creating micro service design. The DevOps does not shows strict methodology in case of development. The only requirement is that the product could be developed independently at parallel with other developers at the same time which was not mentioned in agile, scrum or DCMM as key value.

2.6 Summary

The DCMM management of IT department is worth to use in combination with development management such as scrum, agile development or DevOps. When combination is chosen by a company it will take all of the advantages of each management technique and the IT department is able to deliver the product at best quality. However when combination of techniques is applied there is no strict division where solution is measured as being innovative and successful. Imagine that DCMM is combined with scrum framework. There exists a founder of company who manages the software development as product owner. He has strict vision and the scrum team must deliver it. The founder of the company is not opened in changing the requirements. Is the founder of company going to be successful when he is delivering only what he or she thinks is the best or what was received by users? In the DCMM terminology the open mindedness is key value of the department and so there must be clear instructions how to apply multiple of the software development techniques.

There was not much mentioned about risks. There is always a risk when developing new IT product. Nowadays, security of the products must met very high expectations because of the possible money and reputation loss which results in being unsuccessful. It was slightly mentioned, that changing management is involved in ITSM terminology to evaluate risks. There are also agile techniques how to calculate possible revenues. In this case it has to be IT management that is considering risks which should be discussed in close relation with IT development department. DCMM does not focus on this side of IT project which is also important even though it could be considered as administrative overhead. However I think it should be considered as argument while making discussion about creation of new product.

Chapter 3

Design of information system and analysis of tools API

As it was discussed in previous chapter thesis focuses on DCMM management model where created application will do three main tasks.

1. Event collector from various sources and tools which could be transformed to DCMM activities, stories and components.
2. Classification of gathered events to activities stories and components. Creating connections between stories, embedding stories to another stories and measuring spent resources based on the time consumption of events.
3. Visualisation of the activities, stories and components. Easily readable measured metrics that can be evaluated by IT management.

To fulfil each of the tasks the system will consists of three components. First will be collector of events. As DCMM is IT management tool it will surely be connected to software development tools like Github or Gitlab which are widely used versioning systems[1]. To be able to segregate stories and create stories with activities versioning system is not enough. Stories could be segregated using project tracking software. As single representative was selected Jira, which is very popular in IT community[2]. It allows different types of project management like waterfall software development paradigm, agile software development paradigm or scrum framework which is based on agile principles. DCMM could be applied to any of these. This is minimalist design setup that should be met in order to receive enough IT management information.

In addition there are different sources of information that could be used. However there has to be implemented sophisticated text classifier that is able to create DCMM activities and stories. As other source of information could be used Trello which is ticket creating tool that could be used by IT department. Another one is Google Calendar which allows users to create multiple calendars and separate events to more and less important and add custom description.

[1]Github: `https://github.com/`, Gitlab: `https://about.gitlab.com/`

[2]`https://www.atlassian.com/software/jira`

3.1 Information system design

The DCMM information system design is shown on figure 3.1.

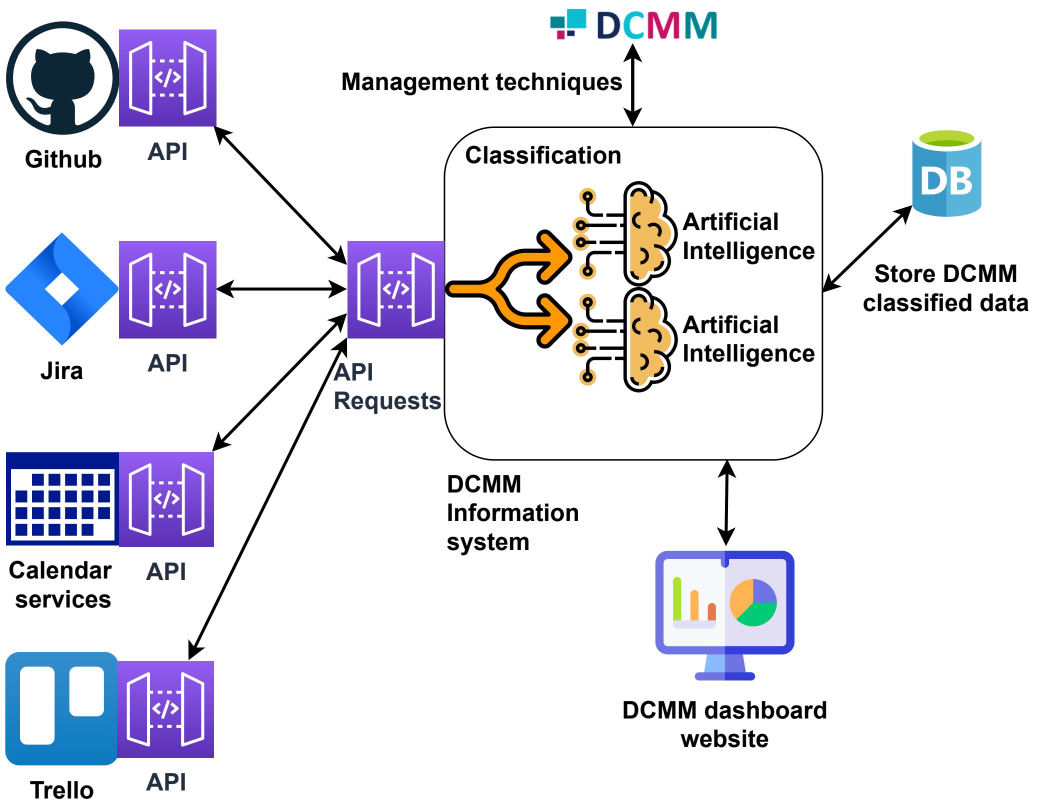


Figure 3.1: DCMM information system.

On the left of the figure are listed tools from which information are gathered. Each tool contains different REST API where information system API requester sends GET messages to obtain all necessary information. As the repositories or project could be also private, there is need to sign in using Oauth sign in which generated API token that is used to gather information. Obtained information are classified with help of artificial intelligence. Classified DCMM activities, stories and components are stored in NoSQL database because single schema cannot cover all of various information fields. It is due to different information gathered and so the classified data could miss some part or are additionally changed. Once the classification is done there exists dashboard web page that is accessible by any user. Both simple and complex visualisation are shown based on contents of the database.

From user interaction perspective user is able to filter shown diagrams, verify correctness of classification, show metrics and create new stories filling relevant information. In order to modify or obtain new data administrator of DCMM Information system can check last synchronization and run synchronization with APIs manually. As the manager is in charge of IT he is able to create and select DCMM components (hardware components, software etc.). These components can be coupled with stories to be able to filter the usage of components.

3.1.1 Class diagram

Designed system consists of singleton Database class, factory method creating gatherer based on the type and facade that specifies execution steps. All of these are shown on figure 3.2.

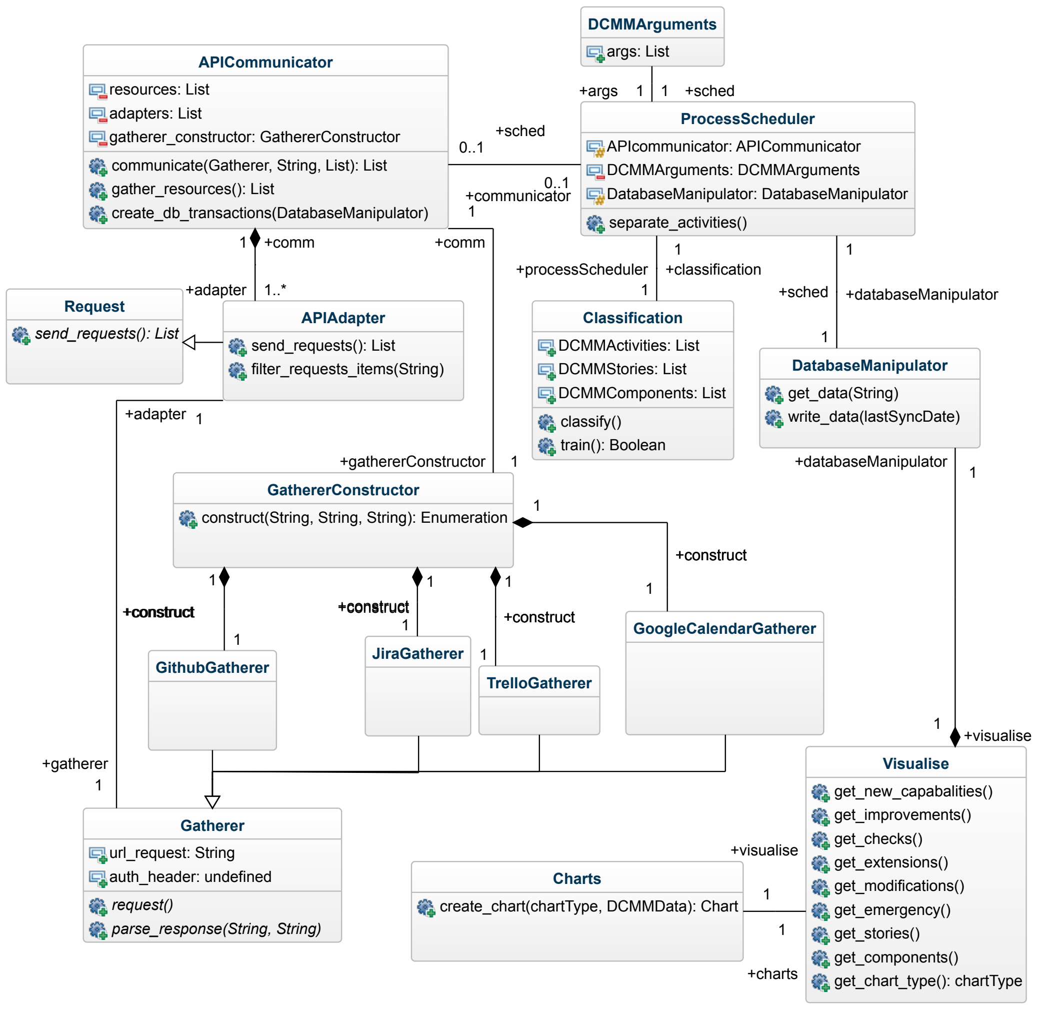


Figure 3.2: Class diagram of designed information system.

Classes are designed based on the DCMM information system schema. On the left side is API communication using adapter that adapts request response to desired output that is analysed further in this work. DCMMArguments class exists because of usage of personal generated tokens that cannot be stored on the system because of confidentiality of the information. They are used single time for HTTP responses and discarded afterwards. Classification part takes Database data where are inputs of DCMM process and classifies them based on the trained models that are discussed in chapter 4. Visualisation gets stored classified data from NoSQL database and put them into Charts class where from DCMM data and char type it generates chart that will be displayed on web interface. Whole web classes are not part of design since this system will use Django web framework that deals with user interface design, authentication and other problems that are not related with classification and gathering of DCMM information [21]. To solve chart problem this work will set basis of dash and plotly framework that generates different charts from data in json format[12]. Thanks to the NoSQL database dense information could be stored, analysed and classified that does not have exact schema. It is due to different responses on different tools that cannot be exactly match together. For the NoSQL database it was selected MongoDB and MongoDB driver for python[10].

3.2 Tools API analysis

In the following sections will be described in details how APIs of selected tools works. Analysis consists of highlighting important parts from DCMM classification perspective.

3.2.1 Github and Gitlab

As first tools was chosen versioning system. This system allows IT developers to contribute on open or closed source project. There exists recommended approach how to use it and how to contribute.

When sending request to Github API all worth information that could be received are listed below in Github API Request.

```
Github API Request:
GET /repos/docker/compose HTTP/1.1
Host: api.github.com
User-Agent: curl/7.58.0
Data:
"id": 15045751,
"name": "compose",
"full_name": "docker/compose",
"private": false,
"owner": {
  "login": "docker",
  "id": 5429470,
  "url": "https://api.github.com/users/docker",
  "html_url": "https://github.com/docker",
  "repos_url": "$owner[url]/repos",
  "events_url": "$owner[url]/events{/privacy}",
  "type": "Organization",
},
"url": "https://api.github.com/repos/docker/compose",
"description": "Define and run multi-container
                applications with Docker",
"tags_url": "https://api.github.com/repos/docker/compose/tags",
"assignees_url": "$url/assignees",
"contributors_url": "$url/contributors",
"commits_url": "$url/commits{/sha}",
"issues_url": "$url/issues{/number}",
"labels_url": "$url/labels",
"pulls_url": "$url/pulls"
```

```
    "created_at": "2013-12-09T11:40:58Z",
    "updated_at": "2020-12-22T06:51:32Z",
    "open_issues": 461,
```

The most valuable are selected URL locations where could be found dense information about ongoing project, how it is solved and what types of DCMM activities are happening in case of managing IT project. The reason behind choosing these attributes is that there has to be simple identification of given repository that forms single project. This project has owner or company name as owner. There are stored URLs for user that owns repository. It is a list of all repositories that the owner owns and events that are upcoming for him. After that information takes place description of repository with URLs to tags (all versions of project e.g v1.0.0). From DCMM perspective there has to be managed HR where developers are also known as contributors. Then all of the commits, issues and labels are forming group of important URLs where it is possible to create DCMM activities and stories from these information. To be able to visualise the issues, they are linked with pulls_url pull requests. There could be specifically assigned people or developers to the repository. These assignee's take care of repository integrity and are responsible for new versions, tags and following recommended guidelines in terms of creating issues solving it in branches etc. To be able to specify from which date should be created DCMM classification there could be selected date boundaries, date of creation and date of last update.

As DCMM classify types of management activities to 10 types, Github has 3 and more representatives for `issues` which could be divided into 5 to 6 of DCMM activity types[11]. This is dependant whether repository uses recommended labeling mechanisms for issues which could separate issues to multiple activities. When new functionality is added to repository in Github language it is called as `feature` issue which is related to DCMM improvement activity (IM) or new component or capability activity(NC). A `bug` or `bugfix` issue is something that has negative impact on resources or both resources and capabilities. It refers in DCMM activity as modification (MO), extension (EX) or wrong doing (WD) activity. It depends whether fixing bug allows user to use functionality. WD issue is related to security fix which could slow down or change behaviour of program to get rid of possible breach. End user does not receive new capability and it costs resources and capabilities are reduced.

Standard labeling mechanism working on versioning systems are described in table 3.1[20].

Label	Description
bug	Indicates an unexpected problem or unintended behavior
documentation	Indicates a need for improvements or additions to documentation
duplicate	Indicates similar issues or pull requests
enhancement	Indicates new feature requests or improvement
good first issue	Indicates a good issue for first-time contributors
help wanted	Indicates that a maintainer wants help on an issue or pull request
invalid	Indicates that an issue or pull request is no longer relevant
question	Indicates that an issue or pull request needs more information
wontfix	Indicates that work won't continue on an issue or pull request

Table 3.1: Standard labeling mechanism which is supported in Github/Gitlab versioning system.

The issues itself have default labeling mechanism guideline. However it can be customized and that is why it has to be supported in some other ways where $ refers to variable.

```
issueDescription -> $text
issueLabel -> bug | documentation | duplicate | enhancement |
              good first issue | help wanted | invalid | question |
              wontfix
issueType -> src | doc | lib | tools | deps | benchmark | test
issueKind -> bug | Bug | BUG | fix | Fix | FIX | BUGFIX
issue -> $issueType: $issueDescription[$issueLabel] |
         [$issueKind]$issueDescription[$issueLabel] |
         $issueDescription[$issueLabel]
```

When repository has own issue board and issue labeling there are going to be used word classification for previously mentioned labels and issues kind. When detected then parsing will change based on the detected patterns for the possible labeling.

While working with issues, there are events happening during solution of issue. Considering only opened issues the most important events are `referenced` and `milestoned`[3]. Referenced event is referencing commit message while pushing to certain repository that issue is related to. This event shows progress of single story by amending the code with commits which are describing the progress. The next event is milestoned where issue is part of new release. All of these issues which are connected with new release are forming bigger story where smaller stories of solving separate issues forms single iteration of developed software. It is important that the issue remains milestoned and no `demilestoned` event occurs. When issue has `pull_request` attribute present it means that it is ready to be merged and story has finished from development perspective. It is possible that in future this story can be amended when bug occurs. Then new issue arise which is successor of this issue.

When milestones are missing, project managers often use Github `projects` in order to keep track what has to be done to finish desired functionality. A simple project contains list of opened and closed issues, where division between closed and sum of closed and opened issues is progress of the project solution.

As next element parsed will be commit messages. They are closely connected with issues. When repository follows recommended guidelines there exists following pattern.

- Each issue has separate branch that refers to issue, ideally with number of issue #number or branch name contains #number of issue. It is where DCMM activity starts.
- Finished branch contains single or multiple commits where last one (contributed most lately) contains issue name to create connection. In DCMM meaning the activity is going to be reviewed where DCMM names it that it is “on hold”.
- Merged branch into production is same as finished DCMM activity.

Mostly multiple commits form a single story where multiple developers works with each other. There could be also irregularities where single commit could be big rework and

[3]`https://docs.github.com/en/free-pro-team@latest/developers/webhooks-and-events/issue-event-types`

changes behaviour completely which forms single story. In future this stories could possibly be extended with MO, EX or WD when issue arise[11]. These activities are related to a single story based on the default branch. There is never strict relation that the branch must start from the newest branch. Branching is one of the most used micro architectural design where developers can work independently and merge their work together based on where they started to work.

As last parameter of DCMM IT department should be keeping the track of all created versions of a single project. When releasing new version of software there is still time where older versions are supported. This process takes time and only possible is to fix new bugs. New features are in already tagged repositories blocked because new capabilities are added in newest version.

Assignees are connected with commits and issues and approvals of merges. These people are dedicated to be responsible for fixing, creating new ideas and approving new capabilities that are added to software.

Case study

To be able to understand single DCMM workflow transformation from Github repository, the description will use open source repositories docker/compose and nodejs/node as examples. They were selected due to different project management possibilities that are offered by Github. As the story can be finished and non progressing from current perspective or currently progressing, from both repositories were selected 2 issues from which one is opened and part of milestone and second is closed and was not part of a milestone. For docker compose repository were selected issues found below and milestone 1.28.0.

```
https://github.com/docker/compose/issues/7919
https://github.com/docker/compose/issues/8000
https://githubo.com/docker/compose/milestone/48
date 29.12.2020
```

From nodejs node repository was selected project `Flakees in CI` and two issues as well as in docker compose scenario that are part of the project.

```
https://github.com/nodejs/node/projects/8
https://github.com/nodejs/node/issues/22006
https://github.com/nodejs/node/issues/20750
```

First issue of `docker compose` is already closed and issue number 8000 is opened at the time of gathering. Following table 3.2 shows story identification based on the information gathered.

First story is milestone, where single iteration will be finished. Github API does not support direct listing of issues in milestone. This is why milestone number should be used as filter for issues. Rest of selected stories are connected with activities 7930 and G2 that are created from events inside issues. After further examination the issue 7919 was not milestoned but is related to 7930. It means that even though issue 7919 was not milestoned it is a pull request so it takes a part in the software development iteration.

When considering naming of activities I chose to call activity by number of pull request connected with issue number. The pull request number can differ, however it is connected with tag of issue number that resolves the issue e.g `Resolves #7919`. Complete output of Github events response example can be found in appendix A.1. In depth analysis of single

Story ID	Story name	Story Navigator	Activities	Date
docker/compose 1.28.0	Milestone	aiordache	8000, 7977, 7971, 7930, ...	2020-09-07T13:11:51Z
docker/compose 7919	Implement service profiles in Compose	chris-cone	7930, M7930, RR7930-1, RR7930-2, ...	2020-11-10T15:31:04Z
docker/compose 8000	Windows: remote docker context via SSH fails on Windows	tero-dev	G2	2020-12-16T19:14:41Z

Table 3.2: Transformed stories from Github docker compose repository.

pull request or issue could be divided into even smaller parts like code review, changes made based on the reviews and further discussion as it can be seen in table 3.3.

Author of DCMM book has strictly created connection between activity and system component. However Github cannot manage system components like hardware or software so the table is missing this column since it will be always empty.

3.2.2 Jira

Jira software can be used to track IT project progress. To be able to extract DCMM activities, stories and components from Jira there has to be filled information about project issues. Same as in Github or Gitlab section the issue can be new feature, enhancement or bug.

To highlight important parts of the issue extraction a simple sample was gathered using free trial Jira license and predefined project. This sample could be find in appendix A.2. An issue consists of assignee that should work on defined issue. Important is to gather all of the users in Jira and create HR database where all employees are listed. For further classification comment section of issue could be used. The timeline of project issue is created based on the `created` component that shows date of creation of issue. The creator has to be specified in order to find responsible person of solving and merging issue. Such creator is only responsible of creating an issue but reporter of the issue is a person that was responsible of reporting or finding the issue. Often this person should resolve issue and setup initial solution and collect all information that assignee needs to resolve the issue.

As the commentaries are going to be used in further analysis and classification same applies for description. A description of issue is what creator thinks of issue and how to solve it. Also there could be included some snippet of code, figure that shows problems or approach to solution. Such element is hard to add to classification process of DCMM activity or story. Each issue has priority that is divided into 5 levels from lowest to highest. The DCMM measures how many resources were spent on solving an issue. In both Jira and DCMM perspective valuable resource is time spent on solving the issue. This time is part of the progress. Once the solution of issue starts its status changes from `To Do` to `In progress` when it is finished it goes to `Done/Resolved`. All the relevant sub tasks can be

Activity ID	Activity name	Assignee	Story ID	Date	Activity type
7930	Implement service profiles	acran(Roman Anasal)	docker/compose 7919	2020-11-15T15:07:54Z	NC
CO7930-1	Implement service profiles	acrant(Roman Anasal)	docker/compose 7919	2020-11-15T15:05:25Z	NC
CO7930-2	Implement service profiles	acrant(Roman Anasal)	docker/compose 7919	2020-11-15T15:07:54Z	MO
CO7930-3	Implement service profiles	acrant(Roman Anasal)	docker/compose 7919	2020-11-16T09:26:21Z	MO
MI7930	Milestoned	aiordache	docker/compose 7919	2020-12-01T15:01:31Z	NS
RR7930-1	Review requested	ulyssessouza	docker/compose 7919	2020-12-02T12:42:40Z	CK
RR7930-2	Review requested	aiordache	docker/compose 7919	2020-12-02T18:56:52Z	CK
ME7930-1	Merge pull request #7930 from acran/profiles	aiordache(Anca Iordache)	docker/compose 7919	2020-12-02T18:57:19Z	NC
CO7930-2	Add documentation for Compose service profiles	acrant(Roman Anasal)	docker/compose 7919	2020-12-03T19:39:29Z	EX

Table 3.3: Activities gathered from docker/compose open source repository.

found under `subtasks` item. A summary of the task should contain short description. In Jira the summary of issue could be connected to Github to particular issue e.g docker/compose repository issue Implement service profiles #7919 could be in Jira summary "[New feature]Implement service profiles #7919".

Each issue has particular issue type where description helps to relate DCMM activity with issue. In Jira agile development there exists 7 categories of issue[1].

1. Sub–task - Sub–task of an issue. In a logged issue, there can be different tasks to resolve it, which are called as sub–tasks.
2. Bug - A problem that impairs or prevents the functions of the product.
3. Epic - A big user story that needs to be broken down into smaller solvable parts.
4. Improvement - An improvement or enhancement to an existing feature or task.
5. New Feature - A new feature of the product. At the time of creation it is not developed.

6. Story - A user story. Meaning of story is the same as in scrum development.

7. Task - A task that needs to be done to achieve team's goal.

A Sub–task of a Task forms chain of tasks that are related to the original task that appeared first. When bug occurs in the system it is related to one of the DCMM activity types RC, ER or WD type[11]. Improvement could be either IM type where it has positive impact on resources or EX where it has negative impact on resources. NC activity type in Jira is related to new feature. Epic is bigger story which is same as chain of stories in DCMM. The same could apply for story or task. However development team should avoid Jira task because it has both DCMM activity and DCMM story perspectives.

3.2.3 Trello

Trello is used to organize different things under user specified labels. This web tool could be used in IT department to sort management activities, development activities and others. It is fully based on the text so it is expected to follow some structure or key words in order to be able to extract DCMM activities and stories.

Trello divides different activities into boards. These boards have lists of related activities that board should cover. Single activity is known in Trello language as card. Every record is text based and has `name` member, date of creation and `memberCreator` who created board, list or card. These records could be differentiate using `type` element in json output from Trello API.

3.2.4 Google Calendar

Calendar is one of the most common used software to keep track with management activities. In the IT department it is not commonly used because IT focuses more on the software or hardware development progress instead of the management activities. However the calendar events are possible to be transformed into the DCMM activities or stories. Digital calendar supports a way to differentiate between multiple groups. It can be visualised as a person that is part of 3 groups having 3 calendars on his desk where each of these calendars has separate events on it. This is called as a list of calendars. In each of this calendar there are events that are formed of date of creation, summary of the event, creator, start and end date of the event. The DCMM activity has to have exact timestamp when it happened and how much time it cost. This is in a case of calendar little bit problematic because people cannot measure time of solving event. In comparison with Jira as it is project tracking software there is exact time measured during solution of project that is filled by developer. Then big story that consists of smaller tasks could be summed into one number how much does the implementation cost. In case of calendar it is worse as the events are setup for strict hours and mostly there is some time offset given to be sure that everything is said during the event. Summing all the events together has to be done by differentiation of end and start time and sum it all over the specified time period.

The second problem that calendar suffers with in terms of the DCMM classification process is that it is text based as Trello. This involves in classification process to use machine learning on natural language and finding of keywords in calendar events as in Trello events.

3.3 Summary

The designed information system for DCMM IT management has clear structure that could be divided into 3 parts.

1. Event collector.
2. Classification of DCMM events from collected events.
3. Visualisation and metric calculation.

Each of the component could be completely separated and run in parallel so the execution of tasks is not so time consuming. Event collecting happens only once upon time or when manual synchronization is being run. Collected events can be classified and inserted into artificial intelligence that process natural language and only some parts of the collected information. The visualisation has to follow DCMM principles of visually seeing problematic parts of IT management and how innovations progress.

Analysis of Githu/Gitlab, Jira, Trello and Google calendar APIs was carried out. Every tool is different has different API and request results differentiate the most. However during analysis were selected fields and entities that are important in DCMM classification process.

Chapter 4

Approaches of classification

In previous chapter were analysed tools that are used as input to classification process. This chapter will outline approaches that solves text based classification of DCMM entities. This classification will be connected with general pseudo algorithm that results in DCMM activities, stories and components where input is obtained from tools APIs in real time. Output of the classification process is a model that uses machine learning algorithm which solves natural language processing (NLP) problem[22].

4.1 Classification using machine learning

The text based descriptions, summary and information will be classified using dataset gathered from open source Github projects and Jira open projects.

4.1.1 Dataset obtaining

The dataset is obtained from following open Jira projects and Github repositories.

```
https://issues.apache.org/jira/secure/BrowseProjects.jspa
?selectedCategory=all&selectedProjectType=all
https://www.jfrog.com/jira/secure/BrowseProjects.jspa
?selectedCategory=all&selectedProjectType=all
https://jira.sakaiproject.org/secure/BrowseProjects.jspa
?selectedCategory=all&selectedProjectType=all
https://jira.mongodb.org/secure/BrowseProjects.jspa
?selectedCategory=all&selectedProjectType=all
https://github.com/docker/compose
https://github.com/nodejs/node
and other popular Github repositories
```

From Jira project tracker were selected Apache software foundation projects, JFrog, Sakai and MongoDB projects. These repositories are forming 70% of the dataset. As a dense information sample were used docker compose and nodejs, vuejs, kubernetes, velero which creates autolabeling of the issues, repositories. Jira and Github have to follow 2 different approaches of labeling and classification of issues and commits. However due to mislabeling there had to be removed issues with wrong labels or missing labels. Hence it is necessary to process obtained data and match them to correct categories. Classification process of input could result theoretically into 6 different categories, however 4 of them are realistic.

- ✓ Jira new feature and Github issues labeled as feature result in DCMM new component or new capability activity.
- ✓ Jira improvement and Github issues labeled as enhancement or improvement request result in DCMM improvement activity.
- ✗ Jira bug that contains crash, failure, exception or error keyword and Github issues labeled as bug, confirmed-bug with same description as Jira bug result in DCMM emergency activity (Impossible to reach with current dataset).
- ✓ Jira test or Github issues labeled as test are falling into DCMM check activity.
- ✓ Jira bug that contains slow, performance, performance issues etc. and Github issues labeled as bug, confirmed-bug with description containing previous keywords are falling into DCMM modification activity category.
- ✗ Jira improvements that contain documentation keyword and Github issues labeled as feature or documentation including documentation keyword or label documentation or doc are falling into DCMM extension activity category (Impossible to reach with current dataset).

Even though DCMM contains 10 activity types, some of them cannot be classified due to missing labeling and no consideration of such activities during current development process. It is very hard to reprocess current dataset because the problem of NLP is much different in specific technology category than some general wisdom written in english. Previous statements should create such categories that the input is expected to fall into one of the categories and correctness of the classification should be higher than 70%. In the case of lower classification correctness the categories have to be rearranged and more data should be supplied from multiple sources.

4.1.2 Partial automatization of dataset gathering

Since dataset has to be gathered as raw input from website of Jira, in Github gathering case the script for automatic gathering was created. This script should gather all of the issues of single Github project. It is possible to extend it on both open source repository and closed source using either username and password or authentication key. This script also gather raw input of issues from website. The problem of Github issues is that the list gathering of issues works more like summary than detailed descriptor of issues. There are used 3th party tools that solves problem of gathering issues using API calls that are then translated into CSV format.

4.1.3 Gathering input pseudo algorithm

Whole classification algorithm could be separated into 3 stages where in first stage input is gathered from specified API as algorithm 1 shows.

Gathered input is written into database and resources are returned after execution. In case of very long input the resources are not returned because of potential high memory management. This input is then compared with created machine learning model.

Algorithm 1: Algorithm of gathering resources

```
Result: Resources
Input: ToolUrls, Database
Output: Resources
1 Resources = [];
2 index = 0;
3 forall Url ∈ ToolUrls do
4     Resources[index] = Send_GET_Request(Url);
5     index = index + 1;
6     Database.write_data(Resources[index]);
7 if (length(Resources) > 100000) then
8     return [];
9 return Resources;
```

4.1.4 Creation of model

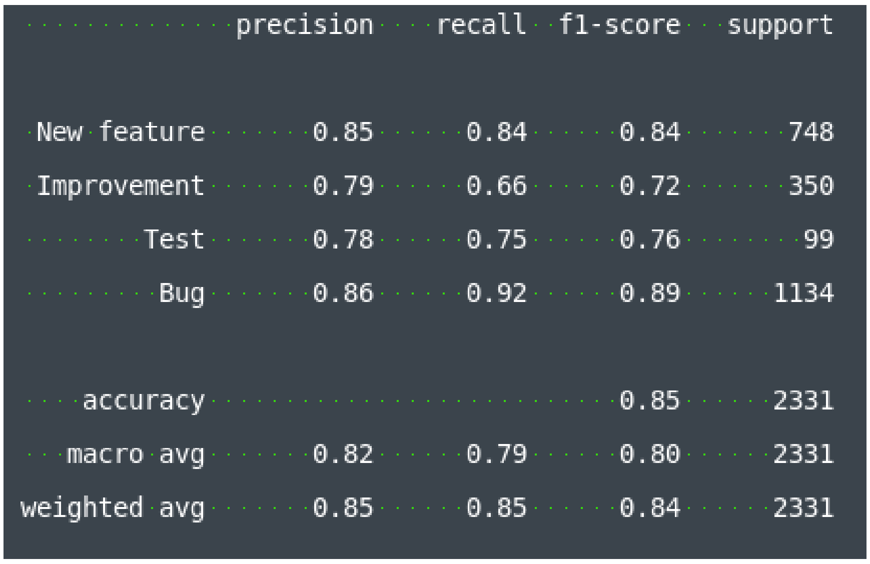
Using prescribed dataset the model that solves NLP problem is created. The dataset considers only issues gathered from Jira and Github, however the gathering and dataset could be easily extended by adding new CSV files with prescribed columns shown in table 4.1.

Fields	Created	Creator	Description	Resolution	Status
Mandatory	✓	✓	✓	✓	✓
Optional	✗	✗	✗	✗	✗
Fields	Summary	Issue Type	Assignee	Labels	Resolved
Mandatory	✓	✓	✗	✗	✗
Optional	✗	✗	✓	✓	✓

Table 4.1: Mandatory and optional fields for dataset CSV file

According to the table, there could be seen that prerequisite are only 7 fields. There are also some optional fields that helps with further visualisation process of DCMM entities. Between the 7 fields are important things like creator of the issue, it's description that is used in machine learning model as input to NLP. Status tells user about current status of issue, whether it is closed, opened or in progress. Summary is short description. The issue type is very sensitive type of information because based on this information whole categorization of NLP is done. When new categories are present in CSV files the model will learn these new categories with associated description and possibly predict that input text falls into this new category. However to add new category there are machine learning metrics that have to be satisfied in order to have correct machine learning model[6, 13]. Current output of these metrics are shown in figure 4.1.

As it can be seen we have 4 rows, `precision` which means in user friendly language how many DCMM entities are correctly classified. `Recall` metric represents that this DCMM class is found in the number of elements of this DCMM class. `F-1` score is mean value between previous two. The last metric is very important and it is `support`. It represents number of occurrences of the given class in selected dataset. Ideally this number should be the same across all of the rows. In this case it could be seen that tests and improvements



```
              precision    recall  f1-score   support


 New feature       0.85      0.84      0.84       748
 Improvement       0.79      0.66      0.72       350
        Test       0.78      0.75      0.76        99
         Bug       0.86      0.92      0.89      1134

    accuracy                           0.85      2331
   macro avg       0.82      0.79      0.80      2331
weighted avg       0.85      0.85      0.84      2331
```

Figure 4.1: Actualized machine learning metrics on current dataset.

are not even near to bugs. It is due to existing management procedures in open source projects. Mainly there are reported bugs in these tools. So the dataset somehow cannot reflect the labeling otherwise. There should be manual relabeling of the bug issues, so the dataset is more even, because some of the bugs are in DCMM perspective wrong doings (WDs) and others are modifications (MOs).

As the dataset was checked and model metrics were measured in comparison with dataset then the actual model is created. There are three steps that should be done in order to create the model:

1. Create converter that strips from Description unnecessary keywords, code snippets and copied strings that could be in previous descriptions. This enhances the model and probability raises much higher than without processing. Also the converter of issue is created that filters wrong labeling of issues.

2. Read CSV file that contains specific project issues. Drop all issues that does not have description and filter them based on the converter rules for issue types and description.

3. Choose training subset from whole dataset. Perform training of this dataset with different experimental parameters. I choosed as reliable machine learning algorithm SGDClassifier[19]. Then select testing sentences that are part of the user's input and perform prediction. Measure mean value between predicted issues and real labeling.

After last step the expected result is that mean value e.g 61% means that model had probability of 61% to guess right label based on the text of testing sentences. However it is not clear because either labeling on sentences side or on the dataset side could be incorrect. So the result is only approximation and the real mean value could be much higher or much lower. To increase the mean value multiple fields and finding of keywords could be helpful in training process. However in this thesis I did not continued further into depth of this classification problematic, but they are clearly outlined in separate chapter.

4.2 Converter approach

Generally converter algorithm is described as algorithm 2.

Algorithm 2: Algorithm of gathering resources

```
Result: Issues converter
Input: issue_types
Output: issue_types
1 setup_issue_types = ['New Feature', 'Improvement', 'Bug', 'Test', 'Enhancement'];
2 forall $label \in issue\_types$ do
3     if $label \notin setup\_issue\_types$ then
4         index = issue_types.find(label);
5         issue_types.remove(index);
6 return issue_types;
```

this algorithm leaves issues that could be found in setup issue list in issue types list that is input. Otherwise the issue is removed from issue types. When single issue does not have any issue type it is filtered by checking for empty issue types value. As it can be seen there is easily extensible machine learning model classification of new categories. By adding new keyword the classification works with new label and shows more categories.

4.3 Choosing training and test sentences

To be able to train the model the training sentences are selected from dataset. I chose to select 70% of dataset, 60% of dataset and 50% of dataset as training sentences. For each of these subset of dataset the evaluation will be performed that will be described in next chapter. To be able to correctly differentiate between test sentence and trained model the input of the model should have even distribution of each of the issue type that it contains. The longer the dataset is the more even distribution of the issue types could be done. However the dataset is not ideal as every project focuses more on improvements (even though the dataset has label of new feature) and bugs and less on actual new features and tests that DCMM book is describing as problem [11]. The classification process can be improved by using implemented solution and gathering from each user input new partial dataset that will be using as feeding for the model. This model will be updated and possible improvement of accuracy could be done.

While considering such dynamic model creation I came across big problem. What if the application model is attacked. The attack is considering the dynamic model changes and the input dataset is unevenly distributing the issues, particular issue description contains different language or random words (e.g Lorem Ipsum). Then the model is rather degrading than improving. This approach could be used only if the input has high filter consists of checks that it should contain only English words, not random words and the issues should be evenly distributed or take constant number of each of the issues.

4.3.1 Filtering sentences

As it was mentioned before the sentences has to be filtered. The basic filter should contain regular expressions that are used for filtering. For such problem there have to be described how *general* issue looks like.

```
JIRA:
SPECIFY ISSUE (HTTP/S LINK TO ISSUE IN OTHER TRACKING APP)
TEXT (could contain HTML/XML tags)
IF BUG:
    SPECIFY ENVIRONMENT SETUP
    IF PROBLEMS:
        SHOW PROBLEMS
{code}
    CODE SEGMENT
{code}
ADDITIONAL ERROR MESSAGES
LONG LOG OUTPUT

GITHUB:
MARKDOWN(ADDED FUNCTIONALITY/ WHAT HAPPENED (BUG))
TEXT (could contain HTML/XML tags and connections)
IF BUG:
    MARKDOWN(EXPECTATIONS)
    TEXT (could contain HTML/XML tags and connections)
    MARKDOWN(REPRODUCTION OF ISSUE)
    TEXT (could contain HTML/XML tags and connections)
ELSE:
    MARKDOWN(WHY TO ADD)
    TEXT (could contain HTML/XML tags and connections)
```

Specification of an issue could have in the worst case sentence that is copied in each issue. TEXT mostly contains some tags or additional information that is dense and does not specify the problem rather makes the classification harder. When BUG is issue type then there is environment setup described in Jira case (similarly in Github case). That is useful when bug is reported because it is showing the NLP that some of these words specifies bug and nothing else. CODE SEGMENT is totally redundant since the application should works across repositories that are developed in different programming languages. Same it goes with LOG output and ERROR MESSAGES. Sometimes are worth to leave them but in most cases are redundant. The language should be clear and contain either solution of problem, new innovative approach or usual keywords that specifies each bug.

The HTTP/S link is always stripped since it has 0 informative value in case of NLP problem. However the link could forward for much more information. However as the collection should take the least possible time and result into covering single tool there were not implemented crawlers for further information according to the links. Also there could be potentially links to the harmful websites.

In case of Github there are popular MARKDOWN tags that are separating important parts of the issues. It is not mandatory, however most of the repositories are using them. This markdown strings that are separating the labels are useful and helps to classify the text correctly.

After examination of the issue structure I chose to perform filtering based on the following regular expressions.

```
STRIP HTTPS => https?:\/\/.*\]
STRIP DEBUG OUTPUT => \[?[A-Z]+[- ]\d+\]?
REMOVE CODE SEGMENT => ^\{code((\||\s|:|=|\.|\#)?(\(.*\))?
                       (\w+|\d+)?)+\}[\s\S]*?\{code\}
REMOVE NO FORMAT SEGMENT => \{noformat((\||\s|:|=|\.|\#)?(\(.*\))?
                            (\w+|\d+)?)+\}[\s\S]*?\{noformat\}
REMOVE QUOTE SEGMENT => \{quote((\||\s|:|=|\.|\#)?(\(.*\))?
                            (\w+|\d+)?)+\}[\s\S]*?\{quote\}
REMOVE COMMENTARIES => ((<!--[\s\S]*?-->) | (‘‘‘[\s\S]*?‘‘‘))
REMOVE DATES => ((\d{4})-(\d{2})-(\d{2})|
                (\d{4})\/(\d{2})\/(\d{2}))
                (\d{2}):(\d{2}):(\d{2})(\.\d+)?
```

These regular expressions should filter most of the input description when it contains unnecessary dense information that has negative impact on machine learning process.

4.4 Summary

The model currently contains 13288 filtered elements. As it can be seen that's not that much as in the case where all companies gave their data together. It would be much much more inside dataset and so the precision would rise and support of each category could be more evenly distributed. This phenomena could not be overdone because of the covid-19 and higher security standards. So the companies are rather closed to giving private information. In case of better propagation that the data could be used for such scientific approach could help the tool analyse the management of the IT project and enhance the management of IT company and save money and time. Given the circumstances, this machine learning approach is the only thing that can be done at present.

Chapter 5

Implementation

The whole system was implemented using `python3` language. The implementation will be divided into three categories as the design is prescribing. Namely the gathering of the events, classification which was mentioned in previous chapter with pseudo algorithms and visualisation and website interface.

5.1 DCMM Gatherer

DCMM gatherer is separate program that works as backend program or standalone package. As it has to be integrated with web application and still be modular the code has to be packaged and installed on the specified system. The installation could be either manually done on the particular system or built using docker which was made for integration testing.

5.1.1 Classification scheduler

Is the main function, where Gathering starts. It consists of following steps that each of the step is in separate subsection.

5.1.2 Input arguments

As there are sensitive information like API keys, usernames and password I chose to not store sensitive data inside database but rather run single application, where all sensitive information like password and API keys are gathered from standard input in secure way using `getpass` function. However to be able to segregate between multiple services there are input arguments that specifies which services are used by the user. User can use single service or multiple services and insert all usernames, API keys and passwords according to prompts. Full list of the command line arguments could be find in appendix**??**.

5.1.3 Constants

To be able to handle different authorization techniques on different services, there exists constants that represents service and credentials format. Following snippet shows possible credentials:

```
# Authorization techniques
NO_CREDENTIALS = {}
TOKEN_CREDENTIALS = {TOKEN_AUTH: ['{token}', '{header}']}
```

```
USER_CREDENTIALS = {USER_AUTH: ['{username}', '{password}',
                                '{header}']}
JSON_CREDENTIALS = {JSON_AUTH: ['{file_path}', '{header}']}
HTTP_CREDENTIALS = {HTTP_AUTH: '{http_path}'}
```

implicit one is no credentials, where service does not need any credentials to authorize. Token credentials are often used by APIs. Each services can generate unique API key for the particular user, where collision could not happen. This token is then used instead of password for particular access and the API matches the generated key with user. This currently suffers because there is no need to specify username that is connected with such token. It easier to enumerate all the keys and wait till authorization process is complete.

User credentials are typically username and password authentication. This is very often done trough API that is representing web page where users sign in with SSL established so the username and password even though is transferred in plain text it is encrypted.

JSON credentials are representing JSON files where authorization token is stored. Company using token credentials is Google. Access to any of the Google application requires to generate JSON token file which allows access to particular service e.g Google calendar service. This file contains client id, authentication provider and actual client secret that is similar as token.

Last possible way to sign into service is by using exact http request, where in options are filled authentication values. This particular way of authentication has disadvantages. Namely the username and password could be easily misused in case of breach in other company and the password and username is same across multiple services. The user cannot check whether he is using the API in web interface so the accounts could be misused to cause harmful attacks without knowing. Using token and username combination prevents such misuse because of 4 reasons[3].

1. Unique – tokens are specific to GitHub and can be generated per use or per device
2. Revocable – tokens can can be individually revoked at any time without needing to update unaffected credentials
3. Limited – tokens can be narrowly scoped to allow only the access necessary for the use case
4. Random – tokens are not subject to the types of dictionary or brute force attempts that simpler passwords that you need to remember or enter regularly might be

Constants also contains gathering unified resource locators (URLs) which are specified by user either based on the combination of input username or location of service. These URLs are connected with particular authentication method and credentials type which was mentioned before. The last constant represents issue types. As it was mentioned DCMM currently differentiate between 4 DCMM entities that are: New Feature, Improvement, Test and Bug.

5.1.4 API communicator

API communicator covers all of the selected and implemented APIs inside one class. It contains general GathererConstructor which takes care of creating particular API gathering class. This part is responsible with invocation of communication with APIs in form of gathering information (resources) and communication or creation of transactions with

NoSQL database. This communicator also has for each gathering class API adapter which translates the information to understandable flow of information that could be used by DCMM machine learning model.

5.1.5 API adapter

General workflow is to adapt the requests to gathering method and parsing of response to parsing gathering method. In case if the parsing is not implemented gatherer gathers all of the information without post processing (adapting) them.

5.1.6 Extension of the API gatherers

In order to extend existing API gatherer or add new one there are 3 things that contributor has to follow.

1. Add name (class name) of the new Gatherer into constants in `API communicator` class.
2. Create Gatherer class e.g BitBucket with constructor with specified credentials and `send_request` method. Optionally `parse_response` method in order to process requested data.
3. Add to constants this new gatherer and it's credentials. Optionally extend the command line arguments, however it is possible also to hard code the constants inside constants module.

5.1.7 Database manipulator

As it was mentioned before, the application works with NoSQL database MongoDB[10]. Python3 has connection with such database using `pymongo` module. In order to have single point to access the database, the Database manipulator class is singleton. As the schema in NoSQL database is missing, the NoSQL database has collections. The program contains these variants of collections.

- model_path collection - always contains elements in tripplets. First element is actual model, second one is vectorizer that is used by the model and 3th thing is tfidf transformer which is also used by the model. The database could possibly hold multiple models and user is able to filter model based on his needs.
- serviceName + username/authentication collection - This collection could possible lead into many different combinations, however it forms single group of collection. Service name is derived from the URL or user defined constant. To segregate between multiple users in real system, each of the authenticated user should have access only to single collection, where collection should always contain client identifier (on website) when multiple users are using single system. For example, collection `Githubtorvalds` would represent collection for Linus Torvalds and others could not match their username with his collection.

Database allows users to get all of their data or filter them based on the creation date, when is present in the issues. Writing of the data is done in batched format that is supported by `pymongo` module and MongoDB. Last operation on database is update query, where database at first does not contain DCMM labeling. This labeling is inserted into field called 'predicted' which is integer value that represents the issue type. Current mapping is as follows.

- New feature = 0
- Improvement = 1
- Test = 2
- Bug = 3

This mapping depends on the machine learning model and dataset gathering order. When bugs are processed first, then the bug would represent number 0 instead of 3.

5.1.8 Prediction

As the data and model are stored inside MongoDB it is possible to predict the input text to particular category. First of all machine learning model takes model, vectorizer and tfidf transformer from database, then input data. As the model had to have filtered issue data, here in prediction there is no need to filter data. However then the possibility of counting mean value between correct and guessed labeling is impossible to done because of missing labels. In this stage it is up to the user whether to use all of the issues or correctly labeled. When everything is finished and labels were filtered, then mean value of the prediction is calculated.

5.2 Summary of DCMM gatherer

The DCMM gatherer module consists of classification scheduler which is start point of the gathering process. It process input arguments that user should input in order to to successfully gather particular repository that is desired. The execution involves API communicator, API adapter that adapts the outputs of particular API response, Database manipulator and machine learning model. The output of the program is single metric, how successful was the prediction based on the correct labels. When incorrect or no labels are present, the gatherer is not able to calculate the success percentage.

5.3 Dataset preparation

This is separate module, that is prerequisite for DCMM gatherer. It creates and saves ML model, vectorizer and tfidf transformer. At start it gathers all of the csv samples, that are stored in constant dictionary called `filepath_dict`. The key of the dictionary is name of repository and value is single string that relates to relative path to single csv file.

As it was mentioned earlier, there are mandatory and optional fields in the csv file. Even tough there is a column `Description` it could be for particular row empty. As the classification process requires this field to be present, all of the empty `Description` rows are filtered. Issue types should be one of the following: Big, New Feature, Improvement and Test. To achieve such labeling, there has to be extraction of keywords from actual issue keys of real issue. This is done trough converter function that assigns one of these labels based on these keywords:

- New Feature = when issue keyword contains substring feature/new-feature/new_feature/new feature.
- Improvement = when issue keyword contains substring enhancement/improvement/-doc/documentation.
- Test = when issue keyword contains substring test.
- Bug = when issue keyword contains substring bug.
- No key = when issue keyword contains none of above.

No key labels are filtered as machine learning model cannot predict correct labeling without assigned label. The prediction is based on the integer value in machine learning model. So all rows of the csv files are concatenated into single dataframe and issue type is factorized using `pandas` module. So the integer values that were mentioned earlier in 5.1.7 are assigned.

At this point the trained sentences, possibly test sentences and their vectors are created using `sklearn` module which is used for working with machine learning models and their vectors[9]. Currently the test sentences are gathered separately also as csv files. Ideally these should be different than trained sentences, so the correctness of prediction is evaluated.

5.3.1 Graphs

To do automated graphs from gathered samples there are mathematical and graphical libraries used on the existing dataset. Concrete graphs are in next chapter 6 which contains experiments. The graphs are created using scattering process of the tfidf transformed values from string vectors and their labels.

5.4 Django web interface

The web application is created using django framework. It consists of general django settings of the webpage and external django modules, URLs of the website and `celery` connector, which is 3th party package for invocation of background tasks in python. Inside this general directory could be found static assets of the web page like CSS files, Javascript files, fonts and images. Then there are also stored HTML templates. There exists a base template of the whole webpage head and body, where other HTML templates could be included. The templates is separated to 4 types:

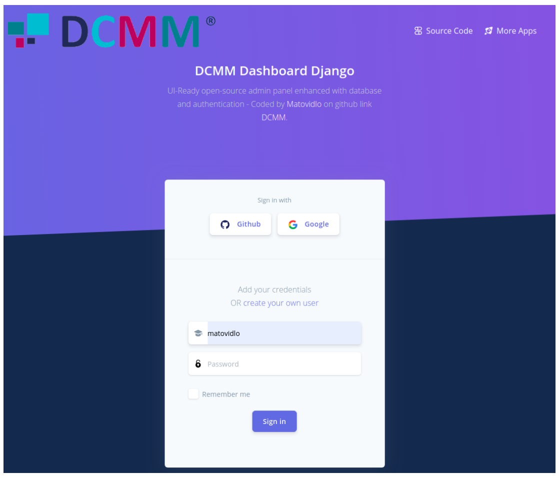
1. Accounts - login, logout and register HTML form for newcommers. In other register case there is also possibility to use social accounts connector of github/jira or other services.
2. Includes - containing header, footer and navigation of the design.
3. Layouts - base template which includes CSS, Javascript files and creates base for each of the HTML URL.
4. Others - concrete views of the application, including page of http error 403,404 and 500.

5.4.1 Authentication

Authentication module takes care of creating new user trough register form or signing the user in or logging out. It is connected with Accounts HTML templates. This module was taken from free django templates with authentication views, urls and models [1].

5.4.2 DCMM

Contains specific urls for DCMM web page. The general workflow is visualised on the figures 5.1 5.1, 5.2, 5.3 and 5.4.



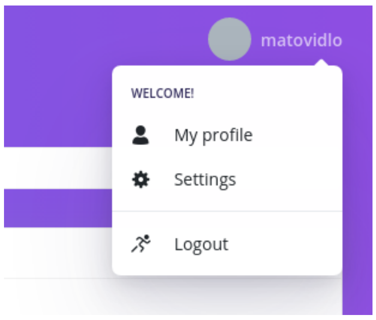
DCMM Dashboard Django

UI-Ready open-source admin panel enhanced with database and authentication - Coded by Matovidlo on github link DCMM.

Add your credentials
OR create your own user

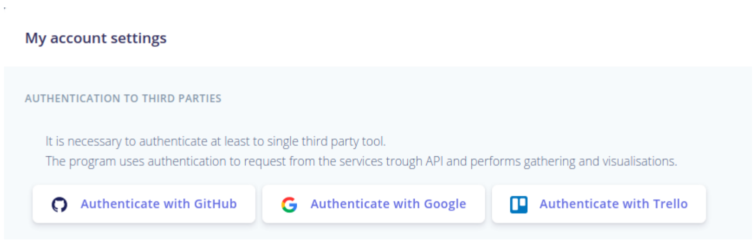
matovidlo
Password

(a) Login web page



(b) Settings of user

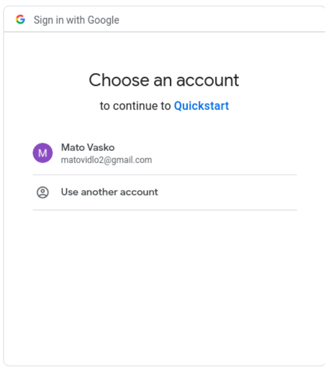
Figure 5.1: DCMM settings workflow using implemented web design


My account settings

AUTHENTICATION TO THIRD PARTIES

It is necessary to authenticate at least to single third party tool.
The program uses authentication to request from the services trough API and performs gathering and visualisations.

(a) Choose one of the service to use DCMM gathering with



Choose an account
to continue to Quickstart

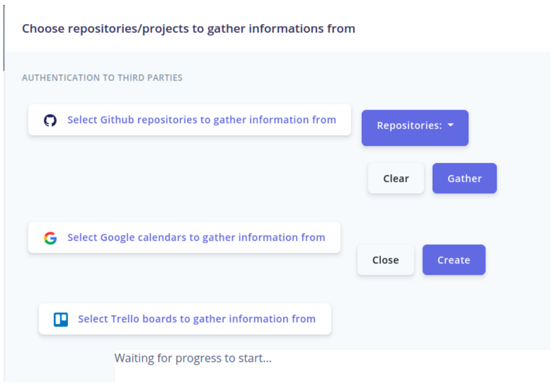
Mato Vasko
matovidlo2@gmail.com

(b) Google authentication was cho-sen

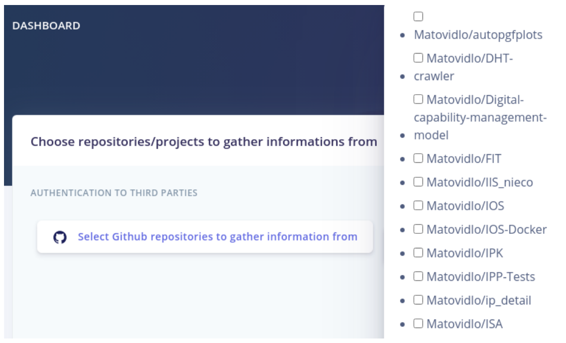
Figure 5.2: DCMM authentication workflow using implemented web design

The concrete DCMM visualisations are part of the next experiments chapter. As it can be seen the designed application is easy to use and no huge knowledge is required by the user. The reason why it has to be easy to use is that it will be used mainly by the management of the IT department and not developers. Even though there could be complains about

[1] `https://www.creative-tim.com/templates/django`


Choose repositories/projects to gather informations from
AUTHENTICATION TO THIRD PARTIES
Select Github repositories to gather information from
Repositories:
Clear
Gather
Select Google calendars to gather information from
Close
Create
Select Trello boards to gather information from
Waiting for progress to start...

(a) Gathering web page


DASHBOARD
Choose repositories/projects to gather informations from
AUTHENTICATION TO THIRD PARTIES
Select Github repositories to gather information from
Matovidlo/autopgfplots
Matovidlo/DHT-crawler
Matovidlo/Digital-capability-management-model
Matovidlo/FIT
Matovidlo/IIS_nieco
Matovidlo/IOS
Matovidlo/IOS-Docker
Matovidlo/IPK
Matovidlo/IPP-Tests
Matovidlo/ip_detail
Matovidlo/ISA

(b) User can choose multiple repositories to gather from

Figure 5.3: DCMM gathering workflow

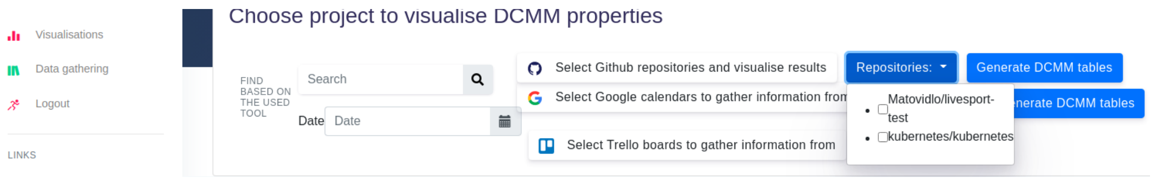

Visualisations
Data gathering
Logout
LINKS
Choose project to visualise DCMM properties
FIND BASED ON THE USED TOOL
Search
Date
Select Github repositories and visualise results
Repositories:
Generate DCMM tables
Select Google calendars to gather information from
Matovidlo/livesport-test
kubernetes/kubernetes
Select Trello boards to gather information from

Figure 5.4: Choosing visualisation of DCMM entities

missing connections with management platforms, there are possibilities to connect it with project tracking platforms but there is no reason why to do so. Such platforms are missing concrete steps of software product development, HR and other things. They are more focused on ongoing project, how it solved, budget assign etc. In case of DCMM the main responsibility lays on developers that actually develops software product that is for sale or for use. So the time could be transferred also to money based on the nominal developer salary.

5.5 Implementation constraints

As the implementation does not reflect whole DCMM logic and book there has to be clarified why it is not covered. The reason lays in phenomena of proving, that classification of DCMM entities are the main part of the work and this application now fulfils the requirement to be proof of concept (PoC) of the DCMM digital transformation of businesses. This business are non flexible companies that solve bugs and not improve their software product or create new ones.

The whole application is missing money management and possibilities to add DCMM components (e.g hardware or any tangibles inside company) that are requisite for the company to count them in whole budget system. The platform solves single workflow, where management would like to know how the ongoing software product is doing and what is the state and what kind of actions are necessary to take to help the product be better than before.

It is possible to extend the whole application for different scenarios and different business however this is minimal implementation that shows the management through the UI and it is able to execute single workflow of gathering and visualisations. These visualisations are then used with combination of DCMM logic to perform next decision in business. The application itself is not strictly telling the user that this is the only right option to take. The actual action is in hands of management as the book is reflecting that fact too[11].

Chapter 6

Experiments

To evaluate the classification model accuracy, program correctness and usability experiments were created and performed. For each of the problem two tests were performed.

6.1 Classification experiment

To better understand problem of classification and data distribution that goes inside the classification process there were created two figures 6.1, 6.2.

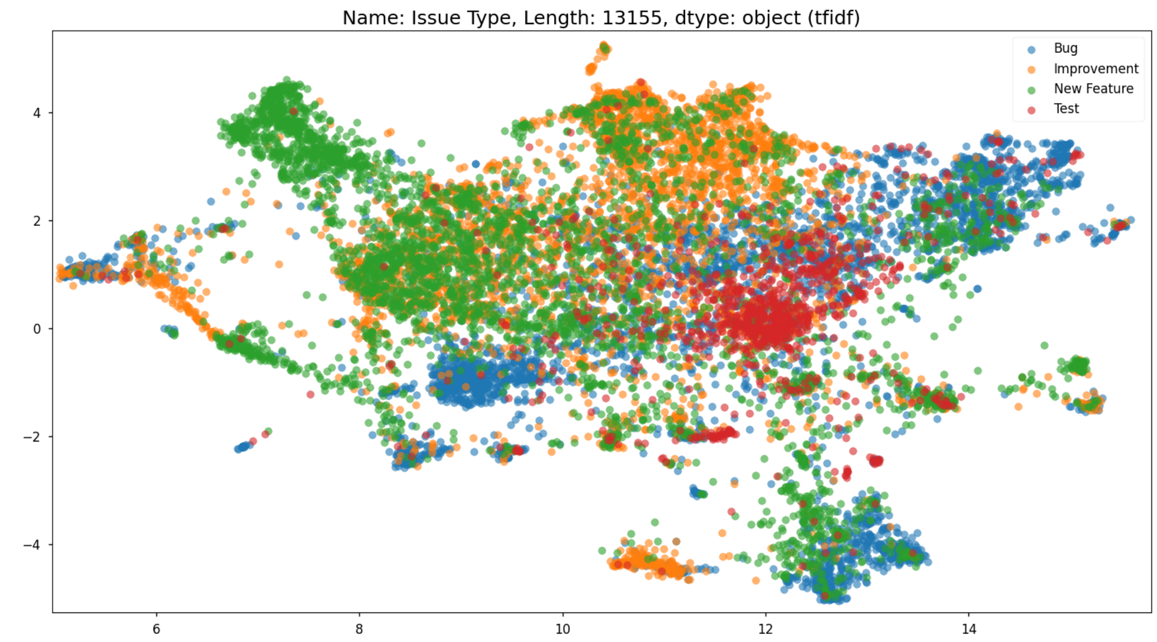


Figure 6.1: Classification with training data set containing 100% of training data.

On this particular figure could be seen that there is quite mess to choose correctly the issue based on the description. There is higher probability to guess that issue is a Bug or a Test but hard to differentiate between Bug and Test or New feature and Improvement. In correct data set the data are separated in almost separate clusters so the machine learning works correctly[5]. Of course the classification depends on input and chosen data set. Different (more accurate) data set could enhance the machine learning and the model would correctly work. With this data set SGDClassifier classifies test data with 85.83% accuracy.

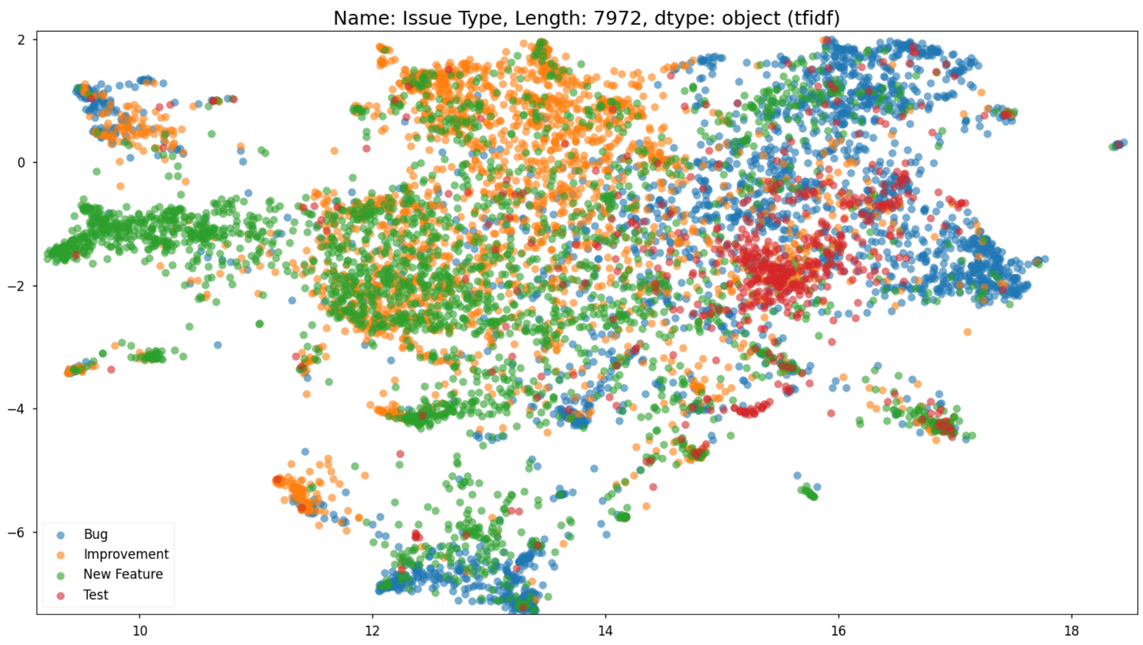


Figure 6.2: Classification with training data set containing 60% of training data.

This particular data set contains only 60% of the training data instead of previous case. Scattered samples contains more spaces between the same labels. This cause incorrectness of classification. Ideally the data forms non separated clusters, so any point that falls into this bigger cluster without gaps or spaces classifies the description correctly. On this particular data set the SGDClassifier classifier the test data with 83.22% accuracy.

6.1.1 Precision

The report contains data sets that are unevenly distributed in terms of classification [13]. This can cause, that final DCMM reports contains more classified data that belongs to classes with high amount of data and less of classes that are in minority inside the data set. However this phenomena could not be changed as most of the open source projects are creating the most issues for Bugs then New Features following with Improvements and the least issues are Test based.

The more precise data are collected and used for classification the more accuracy is correctly diverging to precision that is reflecting the reality.

6.2 Program correctness

To evaluate correctness of created program, the program has to request big amount of data, work with them, store them into database and obtain stored data. As second important factor of the application is to test of concurrently executed gathering tasks. The last examined is to check how the accuracy of the model is changing when adding different repositories with different language and examine the accuracy improvement or degradation.

6.2.1 Large amount of data

The application was able to obtain 27932 issues from open source repositories trough one day and store them into single collection and obtain all of the results and visualise them. 10 github repositories were selected out of the 100 top rated github repositories. The figure 6.3 shows number of issues gathered from the selected repositories.

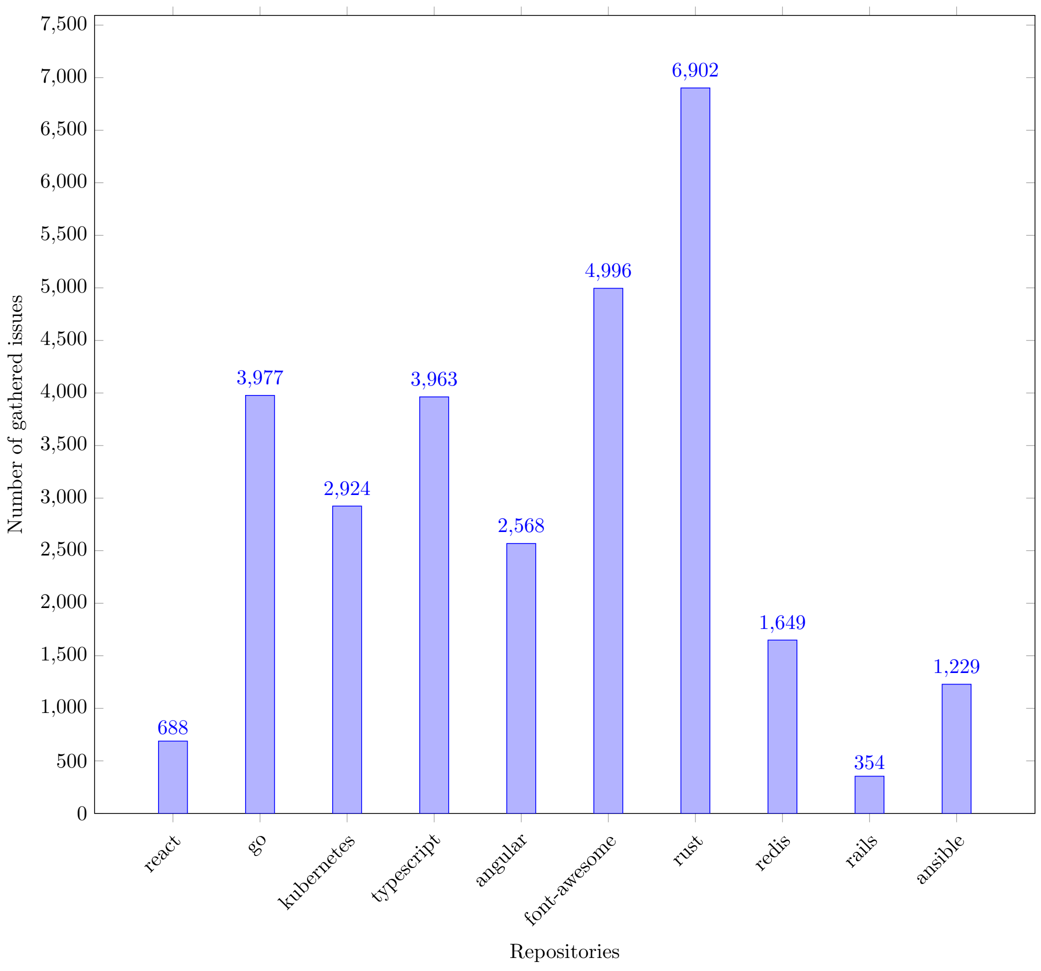


Figure 6.3: Number of issues gathered from selected github repositories

On the figure are 10 selected repositories starting with react. Most of the issues were gathered from rust repository because of the activity of the contributors. The three last repositories did not have that many issues as they are the the least popular in this subset. By further examination of last three, the number of the contributors is much lower than in previous repositories.

6.2.2 Concurrency

The application can be run simultaneously as separate processes. The application itself is not concurrent currently as it is expected that machine will start separate tasks that are concurrent trough third party tool called `celery`[17].

6.2.3 Accuracy and increasing number of samples

It is expected that the accuracy will degrade when more and more samples are gathered from various open source repositories. It is due to different English language used by developers and unevenness of data set inside the created model. There are guidelines in each project, how the issues are called and how to describe the issue itself and assign it correct label. This plays biggest role in accuracy of the repository. The results are on figures 6.4, 6.5.

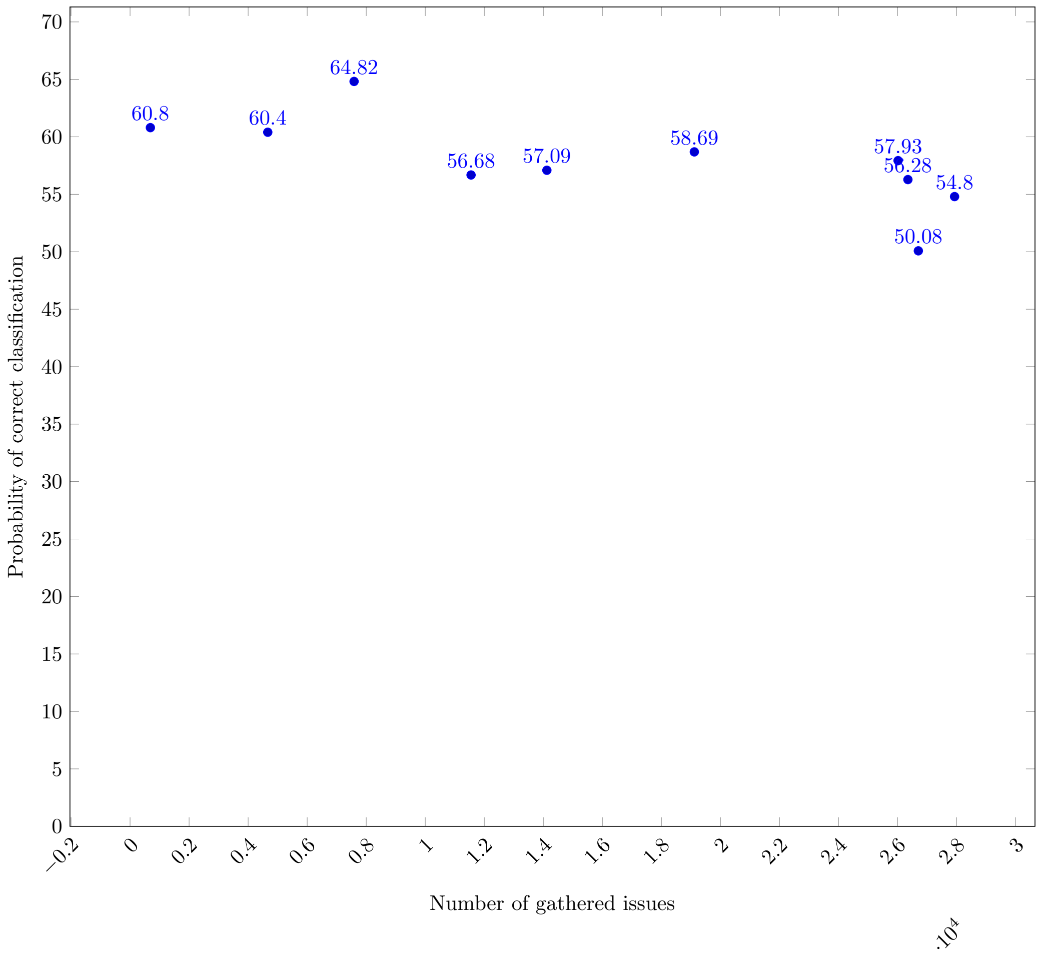


Figure 6.4: Degradation of accuracy in classification process.

Note that the accuracy is degrading over time because of more and more samples are gathered. This is typical for models that are not balanced. Ideally balanced models are not degrading over time and have stable accuracy of classification. However on almost 28 thousand of samples the accuracy is 54.8% which is not the worst result. First 13 thousand samples which is the length of the data set makes the accuracy of the model above 56% on selected data set. It means that the more repositories are used than data set is handling the degradation is even worse and cannot get back above 60%. The conclusion of this figure is that the data set is not equal and does not have enough data. As it was mentioned earlier this is because of not enough public repositories of the companies. Most of the programs and project tracking information are private and cannot be accessed.

Graph on the figure represents labeled issues. The only significant differences are in following repositories in comparison with figure 6.3.

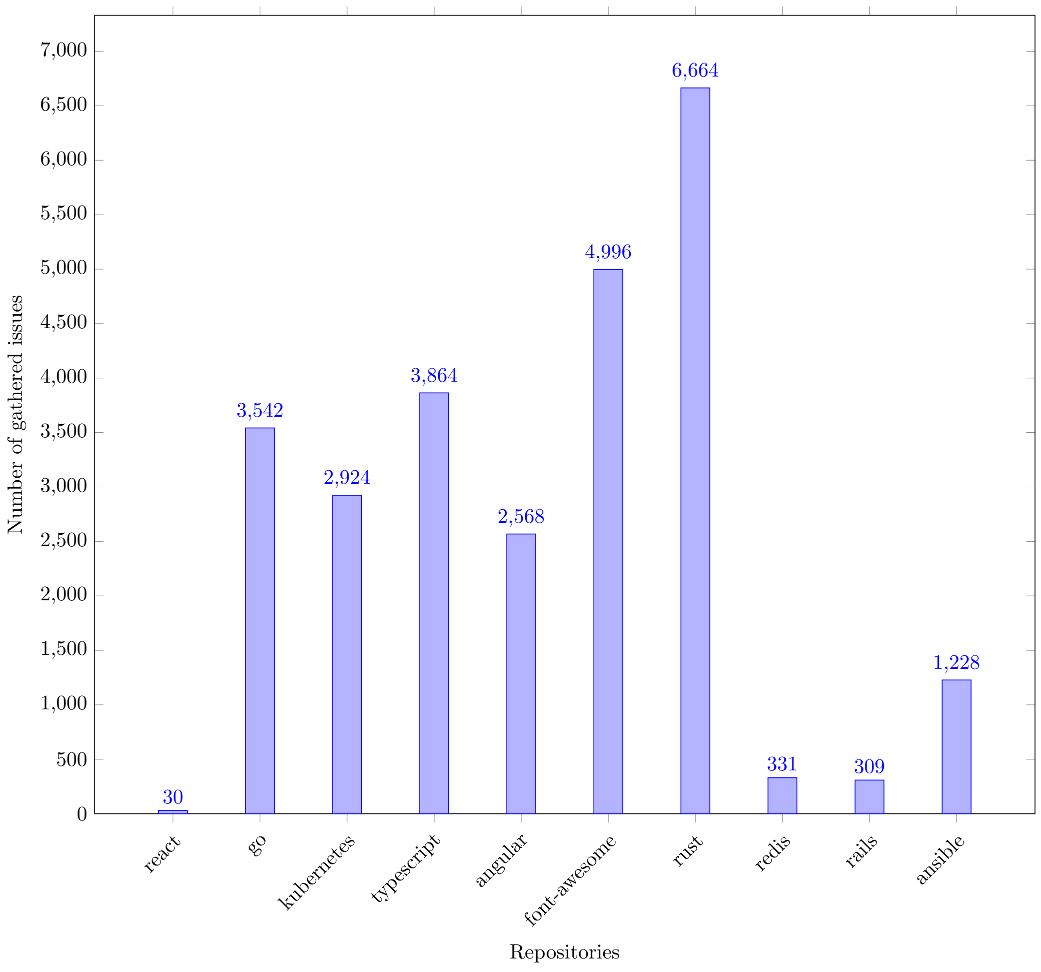


Figure 6.5: Labeled samples in repositories.

- React - labeled issues = 4.36%
- go - labeled issues = 89.06%
- redis - labeled issues = 20.07%

React repository did not helped the model to calculate accuracy, since it is calculated from the label. This means that the accuracy of such repository has very low impact on the correctness of model classification accuracy. This fact should be considered before evaluating the model correctness because the model itself was guessing without having possibility to check whether it was successful. This is also why reinforced learning in this field cannot work because issue contains always description but the assignment of label is done after creation of issue or the label is missing.

6.3 Usability experiments

For usability experiments there has to be taken real world scenario, that describes one particular project in terms of DCMM project management and then the program is used for automatized classification for DCMM.

6.3.1 Case study

For case study I chose the kubernetes Github repository[1]. This repository is big enough to perform such task. It contains on fourth of May 2021 over 1900 issues, 786 tagged versions, 1100 pull requests, 6 ongoing projects, 8 milestones, 3084 contributors and 100336 commits. It is expected that working on such project uses some agile method of development. After checking one of the ongoing project it contains Backlog card so it could be assumed that the project management of kubernetes is either Scrum or Kanban.

In workloads project Backlog currently there are 18 issues labeled as feature, 13 bug issues and 1 documentation. This could tell the management that ongoing project even though the developers are fixing the bugs there are more features than bugs. This is what DCMM focuses on, to be proactive and not reactive.

1.22 is name of the milestone that is the latest. It is expected that this milestone should currently contain more New Feature issues than Bug ones because it is not released and the product should be more enhanced or improved than fixed. Fixing of bugs is expected to be done in earlier released milestones where users are using particular released tags and new problems could occur.

In depth examination of milestone is checking currently opened issues and not overall point of view on the milestone because management wants to know how we are currently doing. In case of whole project preview there has to be made adjustments in the implementation as it is not currently offering such option.

- New Features - 16
- Bug - 23
- Documentation - 2
- Cleanup - 12

This milestone has the oldest issue from 25.10.2018 which is already 2 and half years old and yet not solved. However there is unclassified class `cleanup`. As it can be seen, in this particular case the milestone currently contains more bugs than new features that could look like the project is not that successful in terms of DCMM. On the other hand cleanup issues could play a role in terms of determining DCMM proactive or reactive behaviour in terms of management this product. By manually looking at these 12 issues it could be said that they are improving or adding capability to the kubernetes project. So in based on these issues overall rating of the milestone in current position is **proactive**. To have complete view on the whole milestone, there has to be checked also closed 403 issues.

- New Features - 63
- Bug - 129
- Documentation - 24
- Cleanup - 196

When all of the tags are summed, it is more than 403 labels. It means that some of the issues contains multiple labels. For classification process it could be improvement if it considers multiple labels, however they are not standard ones. The cleanup tag is related

[1]`https://github.com/kubernetes/kubernetes`

in software development either as eradication of previous functionality that should not be supported anymore or refactoring of previously created solution. In this particular tag the sum of features and documentation is not above the number of issues with bug label.

When adding `Cleanup` into account as in previous case the project looks more promising, however cleanup process is either considered in DCMM terminology as extension of digital capabilities or as correction of previous mistakes. Namely the category could be NC, IM, MO or in case of negative impact RC or WD. In case of checking and evaluating each of the issue the machine learning model could be enhanced to determine better the correct label. However for the case study is enough that previous open issues are labeled and overall look of the milestone is **proactive**.

In case of HR of this project, there are these committers:

- liggitt - contributing the most
- andyzhangx - recent contributor
- mtaufen - recent contributor
- MadhavJivrajani - recent contributor
- markusthoemmes - recent contributor
- wangyx1992 - recent contributor

As summary of this Github project, the milestone is evaluated as negative approach to DCMM, because it lacks new features. The project overall in current state is showing the management positively in DCMM terminology, so the management could be satisfied with this project. In terms of HR there is expected that these contributors are showing in network graph that they are contributing and the comparison between them should be for management helpful for potential reevaluation based on their performance.

6.3.2 Program results

To perform case study in created program it contains of three steps, invocation of gathering process on non user repository, signing into Github account trough OAuth application and visualisation of the results

Invocation of gathering process

To obtain these results, first of all it is expected that program is executed with following options to gather from Github kubernetes repository, which is not part of user's repositories but 3th party.

```
python3 classification_scheduler.py --github-username Matovidlo
--token-authentication-github --repositories kubernetes/kubernetes
--nosql-username --nosql-password
```

This invokes the classification scheduler module to use Github username Matovidlo, which refers to my Github account, I would like to authorize with Github token. The repositories is list of repositories that user wants gather information from. For storing the results the NoSQL database is used and authentication is based on username and password. As this is process is repeatable, it ends with current model and current issues with accuracy of **67.52%** on all currently opened issue samples (1856 opened issues).

OAuth sign in

In order to visualise all of the gathered data, first of all there are automated steps to run the service

```
docker-compose up -d redis mongo
```

These two services are requisite for executing the web interface. Mongo refers to MongoDB service and redis is used for gathering process on the website which invokes external scripts trough python module `celery`. Next step is to run website service

```
python3 manage.py runserver
```

As the server is running the user has to register on the website trough simple form. The user has capability of gathering from repositories only when there exists at least single link with 3th party tool like Github/Google calendar or Trello. When a user has one of these links, then the user is able to gather or visualise the data from the service.

Visualisation of results

In case the user performed all of the previous steps, the user is able to visualise the results. It is performed using the view that is on figure 5.4. In this view the user check mark repository `kubernetes/kubernetes` that refers to the case study and the date to visualise data from. For case study purpose chosen date was 01 April 2021. This process ends with showing graphs that are on figures 6.6 and 6.7.

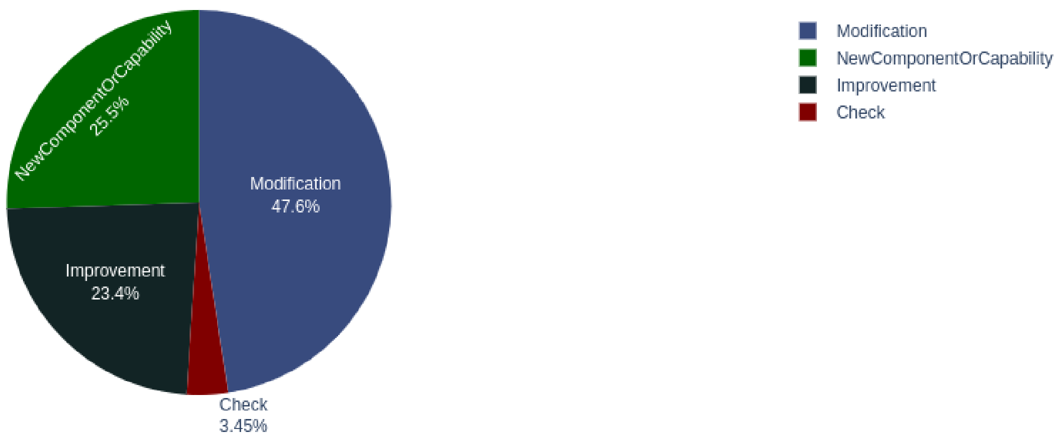


Figure 6.6: Pie chart showing issue types of open issues from April first 2021 till fourth of May 2021

This chart is classic pie chart where each of the guessed issue was visualised trough python module plotly that is used as graph generator.

According to DCMM book this chart should be better and easier to understand by the managers. The more bubbles are larger the more important factor it means. It is important to note, that as the companies should be proactive and not reactive the bubbles that should be big are green and blue and not red one.

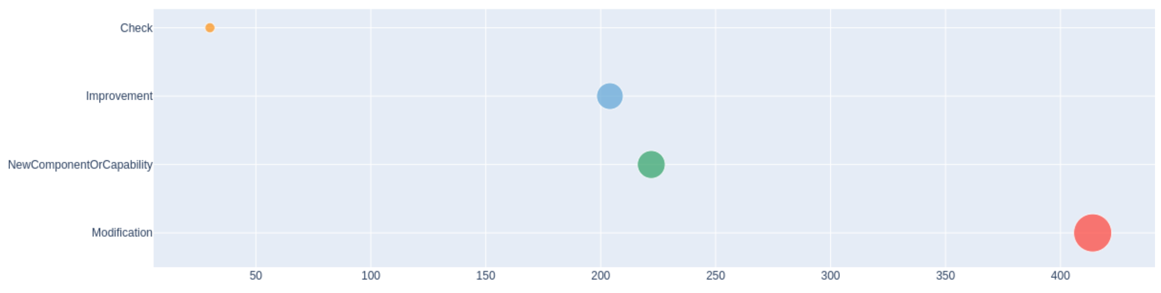


Figure 6.7: Bubble chart showing issue types of open issues from April first 2021 till fourth of May 2021

Next graph covers in the case study evaluation of the milestone 1.22 and others from twenty fifth of October 2018 till fourth of May 2021 that were gathered automatically. The date was selected because of milestone 1.22 oldest issue date. Figures 6.8 and 6.9 shows the milestones and milestone 1.22 in detail.

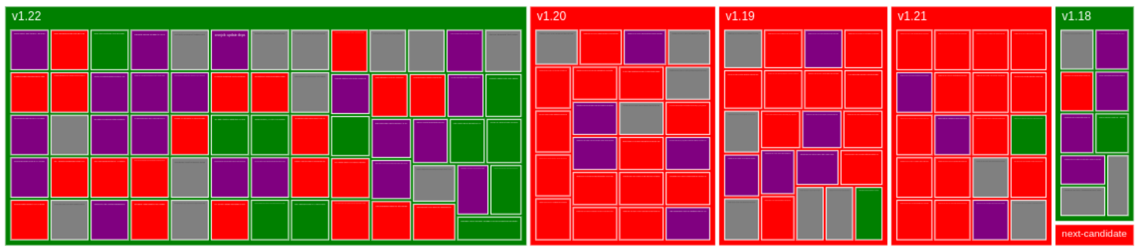


Figure 6.8: Milestones of the kubernetes repository from twenty fifth of October 2018 till fourth of May 2021.

Green rectangles symbol proactive issues New Capabilities or Improvements in the milestone, purple rectangles refer to Tests and red ones refer to bugs or Modifications in terms of DCMM. The gray rectangles are unclassified issue tags by the kubernetes creators. However the machine learning model classified them into one of the 4 existing groups.

Other milestones that were gathered along with milestone 1.22 are also determined. Three of them are fully red, because they are already released tags where it is expected that new features are disabled or not appearing and bugs are fixed.

In detailed view it could be seen that the tag is overall green. It means that there are less bugs than features, so the developers of the milestone are proactively creating new features instead of fixing bugs. In real application there is possibility of hovering over elements, that shows the issue title. There could be visualised additional elements of the issues like creators, number of commentaries and date of creation for denser information for IT managers.

Both of the visualisations refers in DCMM terminology as story, where 1.22 is big story that covers all of the issues. The commentaries, code review and contribution could be visualised inside the issues stories as next story rectangle. As the DCMM is more manage-

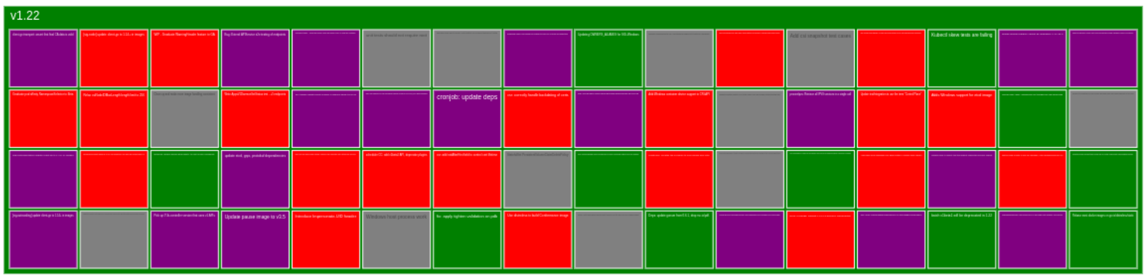


Figure 6.9: Milestone 1.22 of the kubernetes repository from twenty fifth of October 2018 till fourth of May 2021.

ment methodology for IT management, it uses higher abstraction and such granularity of shown information would not be any more benefitial for the managers as current view.

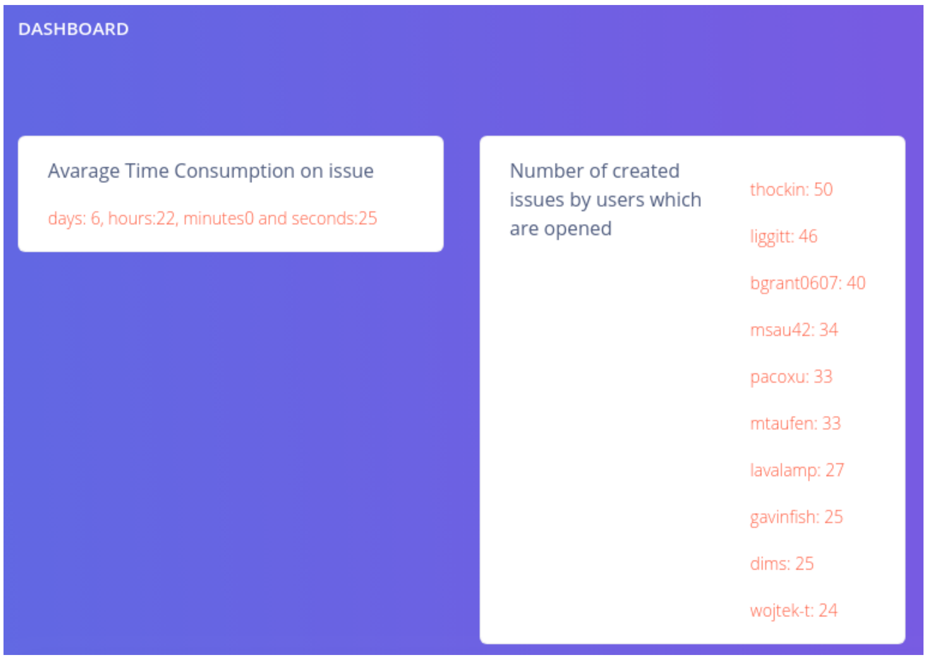
6.4 Usability experiments

Usability experiments are about calculation of different metrics DCMM that plays important role in management process on IT project. According to DCMM there are metrics about resource allocations and distribution of resources between improvements and corrective actions[11]. As the metrics plays huge factor, the dashboard of the application is first page that logged in user see. Dashboard is filled with interesting information about the HR distribution and 10 most valuable persons on the project solution in terms of creating issues and contribution for improvements. The second metrics that is shown is average time to update issue. The data about average time of update of an issue is calculated only on recent updated 500 issues because older issues could not be solved as it could be result of missing technology or prerequisites. This information should be decisive, whether people are solving ongoing issues or the IT is in stagnation process. These metrics are shown in created application on main dashboard at welcome page of the application which is present on figure 6.10.

On the left is mentoined average consumption time on single issue. A single issue in kubernetes case it takes 6 hours to update the state of the single issue. This tells whether IT is doing something or waiting for miracle. On the right side is number of created issues by users. It is expected that those are people that makes the product worth more. These people are often self motivated and like their work.

When the management does not want to check number of issues created by the users, it can also access node network of creators, where sample is on figure 6.11.

On graph the smallest points are issues. Other bigger points are contributors of the repository, where the bigger the point is, the more active the contributor is. This is fast preview of the employees or contributors within the repository to check the motivation of currently employed personnel. For example in kubernetes repository there are 2 biggest contributors colored with purple color that contributed about 20 issues over selected time.


DASHBOARD

Avarage Time Consumption on issue

days: 6, hours:22, minutes0 and seconds:25

Number of created issues by users which are opened

thockin: 50

liggitt: 46

bgrant0607: 40

msau42: 34

pacoxu: 33

mtaufen: 33

lavalamp: 27

gavinfish: 25

dims: 25

wojtek-t: 24

Figure 6.10: DCMM metrics calculated from issues.

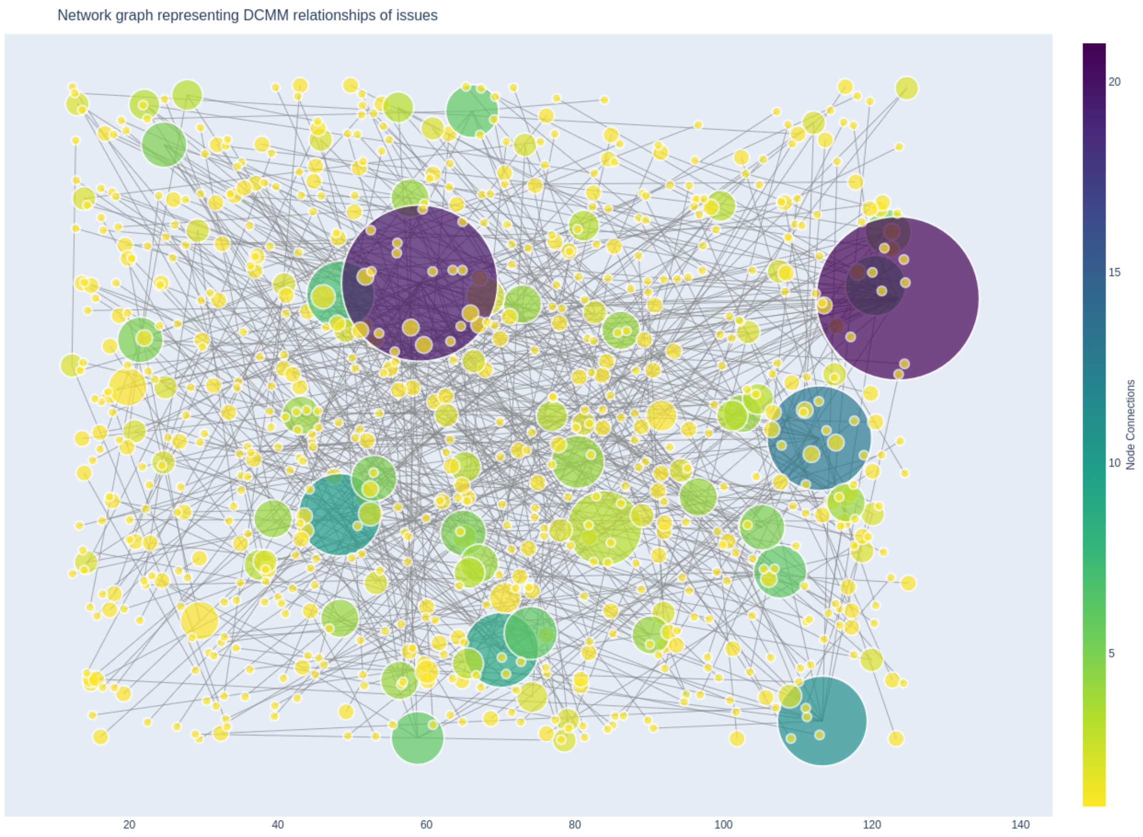


Figure 6.11: Node connection graph of issues and contributors

Chapter 7

Publishing software as open source

As it was from start expected that software will be published as open source, it was decided that it has to have clear structure, ideally good code base that is extensible and well designed interface. For clear structure purpose were used third party packages that have good code base and are widely used by developers. In chapter about implementation were mentioned used design patterns that improves extensibility, usage of APIs that are created by big companies and integration with them, and easily configurable and extensible gathering script and visualisations.

7.1 Github

Selected platform for sharing software as open source was selected Github as it is one of the most used product for versioning the system by developers. Also as the application itself uses this product API to gather information from other open source repositories it is approval that Github was right choice of publishing software. The repository itself contains issues as well as short readme markdown documentation for newcomers. The application is accessible on the url found in the footnote[1].

7.2 Storing secrets

As every application has to follow security criteria, also this developed application had issues with storing secrets. Github nowadays shows the developers that they pushed secrets inside the repository. This has to be remediated as soon as possible, so for this purposes was created `gitingnore` file that contains patterns for ignoring files, where are stored secrets.

In real world deployment, the web application written in django has to disable debug of the deployed web page and generate own secret that is exported trough shell environment on destination endpoint.

7.3 Deployment

Every application that is shared in community should be deployable. In this case the deployment consists of two ways how to deploy the final application:

[1] `https://github.com/Matovidlo/Digital-capability-management-model`

1. Deployment install prerequisite packages and third party tools on running system. The prerequisite packages are inside `requirements.txt` file for python program and for third party tools that should be installed on system in `setup.sh` script. This script install for example mongo database, redis, python etc.

2. Deployment using docker. This is recommended way to deploy the application as it has 0 dependencies that are installed on running system. The `docker-compose` file and `Dockerfile` contains mongo database, redis and application services that automatically deploy. However as prerequisite to run the database correctly in docker environment, there has to be created model by executing following script.

```
python3 dataset_preparation.py
```

This script creates earlier mentioned classification model. This has to be pushed afterwards in docker mongo database trough `insert_model.py` script using following command

```
docker-compose exec mongo python3 /home/insert_model.py
```

This invokes the script inside docker environment that fills model into database. At this point application is running correctly and can be used.

Chapter 8

Conclusion

This thesis was devoted to research of existing approaches of IT management. The concept of DCMM, including its operation, was outlined and compared with existing approaches of the IT management. This comparison showed that DCMM is better approach in management and can be compared to COBIT or ITSM where outcome of using such methodology is higher. On the other hand, it would be very beneficial if DCMM is combined with software development methods such as DevOps, agile or Scrum.

The thesis reflects gained DCMM knowledge to create application that classifies DCMM activities and makes possible monitoring of IT department which consists of 3 components. Created components are event collector that gathers IT management activities from various platforms like Github, Jira etc. The data are gathered using API of the services with user supplied passwords or authentication methods like keys or OAuthTokens. The Application uses simple classification methods and classification using natural language processing where keywords are extracted and used in construction of DCMM entities from gathered data from event collector. The last component of DCMM monitoring platform are visualisations of created DCMM entities. Except of the visualisations there are calculated metrics of the issues and developers of the project that are important in the IT management perspective.

Most of the gathered data are in plain text without specific structure and simple filtering and preprocessing is done. The data are input to neural networks that classifies and detects the meaning of the issues using NLP. The classification could be used on development commentaries, descriptions or summaries of events or all together. Once the data are classified, keywords are highlighted and possibly used in further construction of DCMM entities.

It is known that unstructured data cannot always be correct so the classification can fail. This is unfortunately state that cannot be avoided and must be taken in account. However it is presumed that more structured data as from Github/Gitlab and Jira have higher probability of correct classification of DCMM entities than unstructured data from Google Calendar or Trello. Although both of structured data miss DCMM component. It is an asset that is being used and efficiency or utilization of such DCMM component is evaluated in relation with the time spent on DCMM activities.

The implementation of application is using python language where are two sets of scripts. First set of scripts are devoted to gathering from the APIs of the services. These scripts form one package named `dcmm-gatherer` which is considered as first component of the application workflow that results in DCMM visualisations and guidance. Second workflow is creation of classification model which is part of the `dcmm-gatherer` set of scripts. By invocation of the `dataset_preparation.py` script it results in creation of three binary files that are then by executing another earlier mentioned command stored in database. The last implemented is web service using django framework. The website contains simple registration and login form, dashboard which serves for IT managers as start point for DCMM guidance and visualisations.

The experiments were done in repeatable way, so obtaining the results should be in most cases always the same except of gathering that relies on the current state of the repository. But for forked repository which contains constant issues the results are always the same. The experiments were performed on created machine learning model, where data were scattered into graph to see that the model cannot predict correctly because of the dataset incorrectness. To avoid this, the dataset has to be more balanced. This could be done only, when more repositories and projects would be joined in this thesis. Currently the full dataset contains 13155 of the issues that were labeled and used for classification that has 85.83% accuracy on subset of the data.

The next category of experiments that were performed were usability experiments, where real world repository was selected and examined as case study manually. Comparison of the automatic information gathering and DCMM visualisations and manually gathered information was performed. The automatic gathering reached expected outcomes that were manually detected. The accuracy of the created machine learning model on real world repository that satisfied text format of issue description reached 67.52% accuracy of the detection correct labels using only labeled issues. Unlabeled issues were not examined as it could not be compared. For the DCMM purposes, the data were visualised proportionally which is one of the main focus of representing data in DCMM. Second shown were stories, where biggest story was milestone of the repository, that should be reached. This big story can be colored red or green, where red means that this milestone is not going to extend the digital capabilities of the product. Green color for milestone story is devoted to extension of the digital capabilities when finishing the milestone. The coloring of the milestone is done in simple way, where number of predicted DCMM labels that contains bug are above half from all of the issues inside milestone. This is in a way progressive approach for repository to extend the capabilities.

The last experiment was testing usability of the application by potential IT managers. As the gathering process is finished the dashboard of application shows average time consumption on single issue among recent 500 of them. The second metric is about HR and their busyness. There are shown top 10 contributors that are doing the most work on the issues. The managers have to be familiar with DCMM to understand the importance of the selected metrics. First metric should explain the managers that the issues are rather complicated when the average time is increasing during project solution. The second metric shows that some of the developers or IT members are trying to improve the product over what it is currently. This should tell the managers how hard people work, whether they are motivated and the management is rather progressing to extension of digital capabilities. Otherwise the management should invest into employees or raise their motivation.

The created software solution was published as open source with expectation of further contribution by the community or companies that are interested in DCMM.

Bibliography

[1] Atlassian. *Jira Concepts - Issues.* 2020 [cit. 2021-01-10]. Available at: localhost:8001/secure/ShowConstantsHelp.jspa?decorator=popup#IssueTypes.

[2] Atlassian. *What is IT Service Management (ITSM)?* [online]. 2020 [cit. 2020-12-21]. Available at: https://www.atlassian.com/itsm.

[3] Balter, B. *Token authentication requirements for API and Git operations* [online]. 2020 [cit. 2021-04-03]. Available at: https://github.blog/2020-07-30-token-authentication-requirements-for-api-and-git-operations/.

[4] Beck, K., Beedle, M. et al. *Principles behind the Agile Manifesto* [online]. 2002 [cit. 2020-12-12]. Available at: https://agilemanifesto.org/principles.html.

[5] Brownlee, J. *A Gentle Introduction to Imbalanced Classification* [online]. 2019 [cit. 2021-05-14]. Available at: https://machinelearningmastery.com/what-is-imbalanced-classification/.

[6] Brownlee, J. *Metrics To Evaluate Machine Learning Algorithms in Python* [online]. 2020 [cit. 2021-05-04]. Available at: https://machinelearningmastery.com/metrics-evaluate-machine-learning-algorithms-python/.

[7] De Haes, S., Van Grembergen, W., Joshi, A. and Huygh, T. COBIT as a Framework for Enterprise Governance of IT. In: *Enterprise governance of information technology.* Springer, 2020, p. 125–162.

[8] Fang, A. *Case Study: A Monolithic Software Refactoring* [online]. 2018 [cit. 2020-11-24]. Available at: https://medium.com/shopback-engineering/case-study-a-monolithic-software-refactoring-787b8314a2e0.

[9] Jordan, M. I. and Mitchell, T. M. Machine learning: Trends, perspectives, and prospects. *Science (New York, N.Y.).* 2015, vol. 349, no. 6245, p. 255–260. ISSN 00368075. Available at: http://search.proquest.com/docview/1697220242/.

[10] KS, A. *Working with MongoDB* [online]. 2019 [cit. 2021-01-09]. Available at: https://medium.com/@ashiqgiga07/working-with-mongodb-afee14557a92.

[11] Kvapil, Z. and Boyd, J. *Digital Capabilities Management Model.* Q4IT Czech Republic, 2020. ISBN 978-1723571923.

[12] Moffitt, C. *Taking Another Look at Plotly* [online]. 2020 [cit. 2021-01-09]. Available at: https://pbpython.com/plotly-look.html.

[13] Muthukrishnan. *Understanding the Classification report through sklearn* [online]. 2018 [cit. 2021-04-16]. Available at: `https://muthu.co/understanding-the-classification-report-in-sklearn/`.

[14] Ravichandran, A., Taylor, K. and Waterhouse, P. *Devops for digital leaders: Reignite business with a modern devops-enabled software factory.* Springer Nature, 2016. ISBN 978-1-4842-1842-6.

[15] Richardson, C. *Pattern: Microservice Architecture* [online]. 2018 [cit. 2020-11-26]. Available at: `https://microservices.io/patterns/microservices.html`.

[16] Schwaber, K. and Sutherland, J. *The 2020 Scrum Guide* [online]. 2020 [cit. 2020-12-02]. Available at: `https://www.scrumguides.org/scrum-guide.html`.

[17] Solem, A. *Introduction to Celery* [online]. 2018 [cit. 2021-05-05]. Available at: `https://docs.celeryproject.org/en/stable/getting-started/introduction.html`.

[18] Sotnikov, I. *How to Perform IT Risk Assessment* [online]. 2018 [cit. 2020-12-21]. Available at: `https://blog.netwrix.com/2018/01/16/how-to-perform-it-risk-assessment/`.

[19] Srinivasan, A. V. *Stochastic Gradient Descent — Clearly Explained !!* [online]. 2019 [cit. 2021-03-29]. Available at: `https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31`.

[20] GITHUB INC.. *Managing labels* [online]. 2021 [cit. 2021-01-09]. Available at: `https://docs.github.com/en/free-pro-team@latest/github/managing-your-work-on-github/managing-labels`.

[21] MDN CONTRIBUTORS. *Django introduction* [online]. 2020 [cit. 2021-01-11]. Available at: `https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction`.

[22] Zong, Z. and Hong, C. On Application of Natural Language Processing in Machine Translation. In: *2018 3rd International Conference on Mechanical, Control and Computer Engineering (ICMCCE).* 2018, p. 506–510. DOI: 10.1109/ICMCCE.2018.00112.

Appendix A

Selected tools API outputs

A.1 Github docker/compose issue events response

This is output of issue 8000 events without subscriptions url and received events url.

```
[
{
"id": 4121337588,
"node_id": "MDEyOkxhYmVsZWRFdmVudDQxMjEzMzc1ODg=",
"url": "https://api.github.com/repos/docker/compose/issues/events/4121337588",
"actor": {
  "login": "tero-dev",
  "id": 31454692,
  "node_id": "MDQ6VXNlcjMxNDU0Njky",
  "avatar_url": "https://avatars1.githubusercontent.com/u/31454692?v=4",
  "gravatar_id": "",
  "url": "https://api.github.com/users/tero-dev",
  "html_url": "https://github.com/tero-dev",
  "followers_url": "https://api.github.com/users/tero-dev/followers",
  "following_url": "https://api.github.com/users/tero-dev/following",
  "gists_url": "https://api.github.com/users/tero-dev/gists{/gist_id}",
  "starred_url": "https://api.github.com/users/tero-dev/starred{/owner}",
  "organizations_url": "https://api.github.com/users/tero-dev/orgs",
  "repos_url": "https://api.github.com/users/tero-dev/repos",
  "events_url": "https://api.github.com/users/tero-dev/events{/privacy}",
  "type": "User",
  "site_admin": false
},
"event": "labeled",
"commit_id": null,
"commit_url": null,
"created_at": "2020-12-16T19:14:41Z",
"label": {
  "name": "kind/bug",
  "color": "fc2929"
},
"performed_via_github_app": null
},
{
"id": 4121388670,
"node_id": "MDE0Ok1lbnRpb25lZEV2ZW50NDEyMTM4ODY3MA==",
"url": "https://api.github.com/repos/docker/compose/issues/events/4121388670",
"actor": {
  "login": "aiordache",
  "id": 809903,
  "node_id": "MDQ6VXNlcjgwOTkwMw==",
  "avatar_url": "https://avatars3.githubusercontent.com/u/809903?v=4",
  "gravatar_id": "",
  "url": "https://api.github.com/users/aiordache",
  "html_url": "https://github.com/aiordache",
  "followers_url": "https://api.github.com/users/aiordache/followers",
  "following_url": "https://api.github.com/users/aiordache/following",
  "gists_url": "https://api.github.com/users/aiordache/gists{/gist_id}",
  "starred_url": "https://api.github.com/users/aiordache/starred{/owner}",
  "organizations_url": "https://api.github.com/users/aiordache/orgs",
  "repos_url": "https://api.github.com/users/aiordache/repos",
  "events_url": "https://api.github.com/users/aiordache/events{/privacy}",
  "type": "User",
  "site_admin": false
},
"event": "mentioned",
"commit_id": null,
"commit_url": null,
"created_at": "2020-12-16T19:27:51Z",
"performed_via_github_app": null
},
{
"id": 4121388674,
"node_id": "MDE1OlN1YnNjcmliZWRFdmVudDQxMjEzODg2NzQ=",
"url": "https://api.github.com/repos/docker/compose/issues/events/4121388674",
"actor": {
  "login": "aiordache",
  "id": 809903,
  "node_id": "MDQ6VXNlcjgwOTkwMw==",
  "avatar_url": "https://avatars3.githubusercontent.com/u/809903?v=4",
  "gravatar_id": "",
  "url": "https://api.github.com/users/aiordache",
  "html_url": "https://github.com/aiordache",
  "followers_url": "https://api.github.com/users/aiordache/followers",
  "following_url": "https://api.github.com/users/aiordache/following",
  "gists_url": "https://api.github.com/users/aiordache/gists{/gist_id}",
  "starred_url": "https://api.github.com/users/aiordache/starred{/owner}",
  "organizations_url": "https://api.github.com/users/aiordache/orgs",
  "repos_url": "https://api.github.com/users/aiordache/repos",
  "events_url": "https://api.github.com/users/aiordache/events{/privacy}",
  "type": "User",
  "site_admin": false
},
"event": "subscribed",
"commit_id": null,
"commit_url": null,
"created_at": "2020-12-16T19:27:51Z",
"performed_via_github_app": null
},
{
"id": 4124530442,
"node_id": "MDE0Ok1lbnRpb25lZEV2ZW50NDEyNDUzMDQ0Mg==",
"url": "https://api.github.com/repos/docker/compose/issues/events/4124530442",
"actor": {
  "login": "tero-dev",
  "id": 31454692,
  "node_id": "MDQ6VXNlcjMxNDU0Njky",
  "avatar_url": "https://avatars1.githubusercontent.com/u/31454692?v=4",
  "gravatar_id": "",
  "url": "https://api.github.com/users/tero-dev",
  "html_url": "https://github.com/tero-dev",
  "followers_url": "https://api.github.com/users/tero-dev/followers",
  "following_url": "https://api.github.com/users/tero-dev/following",
  "gists_url": "https://api.github.com/users/tero-dev/gists{/gist_id}",
  "starred_url": "https://api.github.com/users/tero-dev/starred{/owner}",
  "organizations_url": "https://api.github.com/users/tero-dev/orgs",
  "repos_url": "https://api.github.com/users/tero-dev/repos",
  "events_url": "https://api.github.com/users/tero-dev/events{/privacy}",
  "type": "User",
  "site_admin": false
},
"event": "mentioned",
"commit_id": null,
"commit_url": null,
"created_at": "2020-12-17T11:02:58Z",
"performed_via_github_app": null
},
{
"id": 4124530460,
"node_id": "MDE1OlN1YnNjcmliZWRFdmVudDQxMjQ1MzA0NjA=",
"url": "https://api.github.com/repos/docker/compose/issues/events/4124530460",
"actor": {
  "login": "tero-dev",
  "id": 31454692,
  "node_id": "MDQ6VXNlcjMxNDU0Njky",
  "avatar_url": "https://avatars1.githubusercontent.com/u/31454692?v=4",
  "gravatar_id": "",
    "url": "https://api.github.com/users/tero-dev",
    "html_url": "https://github.com/tero-dev",
    "followers_url": "https://api.github.com/users/tero-dev/followers",
    "following_url": "https://api.github.com/users/tero-dev/following",
    "gists_url": "https://api.github.com/users/tero-dev/gists{/gist_id}",
    "starred_url": "https://api.github.com/users/tero-dev/starred{/owner}",
    "organizations_url": "https://api.github.com/users/tero-dev/orgs",
    "repos_url": "https://api.github.com/users/tero-dev/repos",
    "events_url": "https://api.github.com/users/tero-dev/events{/privacy}",
    "type": "User",
    "site_admin": false
  },
  "event": "subscribed",
  "commit_id": null,
  "commit_url": null,
  "created_at": "2020-12-17T11:02:58Z",
  "performed_via_github_app": null
  },
  {
  "id": 4124532343,
  "node_id": "MDE1Ok1pbGVzdG9uZWRFdmVudDQxMjQ1MzIzNDM=",
  "url": "https://api.github.com/repos/docker/compose/issues/events/4124532343",
  "actor": {
    "login": "aiordache",
    "id": 809903,
    "node_id": "MDQ6VXNlcjgwOTkwMw==",
    "avatar_url": "https://avatars3.githubusercontent.com/u/809903?v=4",
    "gravatar_id": "",
    "url": "https://api.github.com/users/aiordache",
    "html_url": "https://github.com/aiordache",
    "followers_url": "https://api.github.com/users/aiordache/followers",
    "following_url": "https://api.github.com/users/aiordache/following",
    "gists_url": "https://api.github.com/users/aiordache/gists{/gist_id}",
    "starred_url": "https://api.github.com/users/aiordache/starred{/owner}",
    "organizations_url": "https://api.github.com/users/aiordache/orgs",
    "repos_url": "https://api.github.com/users/aiordache/repos",
    "events_url": "https://api.github.com/users/aiordache/events{/privacy}",
    "type": "User",
    "site_admin": false
  },
  "event": "milestoned",
  "commit_id": null,
  "commit_url": null,
  "created_at": "2020-12-17T11:03:25Z",
  "milestone": {
    "title": "1.28.0"
  },
  "performed_via_github_app": null
```

```
},
{
"id": 4124532879,
"node_id": "MDEzOkFzc2lnbmVkRXZlbnQ0MTI0NTMyODc5",
"url": "https://api.github.com/repos/docker/compose/issues/events/4124532879",
"actor": {
  "login": "aiordache",
  "id": 809903,
  "node_id": "MDQ6VXNlcjgwOTkwMw==",
  "avatar_url": "https://avatars3.githubusercontent.com/u/809903?v=4",
  "gravatar_id": "",
  "url": "https://api.github.com/users/aiordache",
  "html_url": "https://github.com/aiordache",
  "followers_url": "https://api.github.com/users/aiordache/followers",
  "following_url": "https://api.github.com/users/aiordache/following",
  "gists_url": "https://api.github.com/users/aiordache/gists{/gist_id}",
  "starred_url": "https://api.github.com/users/aiordache/starred{/owner}",
  "organizations_url": "https://api.github.com/users/aiordache/orgs",
  "repos_url": "https://api.github.com/users/aiordache/repos",
  "events_url": "https://api.github.com/users/aiordache/events{/privacy}",
  "type": "User",
  "site_admin": false
},
"event": "assigned",
"commit_id": null,
"commit_url": null,
"created_at": "2020-12-17T11:03:34Z",
"assignee": {
  "login": "aiordache",
  "id": 809903,
  "node_id": "MDQ6VXNlcjgwOTkwMw==",
  "avatar_url": "https://avatars3.githubusercontent.com/u/809903?v=4",
  "gravatar_id": "",
  "url": "https://api.github.com/users/aiordache",
  "html_url": "https://github.com/aiordache",
  "followers_url": "https://api.github.com/users/aiordache/followers",
  "following_url": "https://api.github.com/users/aiordache/following",
  "gists_url": "https://api.github.com/users/aiordache/gists{/gist_id}",
  "starred_url": "https://api.github.com/users/aiordache/starred{/owner}",
  "organizations_url": "https://api.github.com/users/aiordache/orgs",
  "repos_url": "https://api.github.com/users/aiordache/repos",
  "events_url": "https://api.github.com/users/aiordache/events{/privacy}",
  "type": "User",
  "site_admin": false
},
"assigner": {
  "login": "aiordache",
  "id": 809903,
    "node_id": "MDQ6VXNlcjgwOTkwMw==",
    "avatar_url": "https://avatars3.githubusercontent.com/u/809903?v=4",
    "gravatar_id": "",
    "url": "https://api.github.com/users/aiordache",
    "html_url": "https://github.com/aiordache",
    "followers_url": "https://api.github.com/users/aiordache/followers",
    "following_url": "https://api.github.com/users/aiordache/following",
    "gists_url": "https://api.github.com/users/aiordache/gists{/gist_id}",
    "starred_url": "https://api.github.com/users/aiordache/starred{/owner}",
    "organizations_url": "https://api.github.com/users/aiordache/orgs",
    "repos_url": "https://api.github.com/users/aiordache/repos",
    "events_url": "https://api.github.com/users/aiordache/events{/privacy}",
    "type": "User",
    "site_admin": false
  },
  "performed_via_github_app": null
  }
  ]
```

A.2 Jira example project issue output

Simple sample output of single issue in predefined project.

```
{
"expand": "renderedFields,names,schema,operations,editmeta,
changelog,versionedRepresentations",
"fields": {
    "aggregateprogress": {
        "progress": 0,
        "total": 0
    },
    "aggregatetimeestimate": null,
    "aggregatetimeoriginalestimate": null,
    "aggregatetimespent": null,
    "assignee": {
        "active": true,
        "avatarUrls": {
            "16x16": "https://www.gravatar.com/avatar/
            e3bc3332c1a7f7b69b60fc77769f5f06?d=mm&s=16",
            "24x24": "https://www.gravatar.com/avatar/
            e3bc3332c1a7f7b69b60fc77769f5f06?d=mm&s=24",
            "32x32": "https://www.gravatar.com/avatar/
            e3bc3332c1a7f7b69b60fc77769f5f06?d=mm&s=32",
            "48x48": "https://www.gravatar.com/avatar/
            e3bc3332c1a7f7b69b60fc77769f5f06?d=mm&s=48"
        },
        "displayName": "M V",
        "emailAddress": "matovidlo2@gmail.com",
        "key": "JIRAUSER10000",
        "name": "test",
        "self": "http://localhost:8001/rest/api/2/user?username=test",
        "timeZone": "Etc/UTC"
    },
    "attachment": [
        {
            "author": {
                "active": true,
                "avatarUrls": {
                    "16x16": "https://www.gravatar.com/avatar/
                    e3bc3332c1a7f7b69b60fc77769f5f06?d=mm&s=16",
                    "24x24": "https://www.gravatar.com/avatar/
                    e3bc3332c1a7f7b69b60fc77769f5f06?d=mm&s=24",
                    "32x32": "https://www.gravatar.com/avatar/
                    e3bc3332c1a7f7b69b60fc77769f5f06?d=mm&s=32",
                    "48x48": "https://www.gravatar.com/avatar/
                    e3bc3332c1a7f7b69b60fc77769f5f06?d=mm&s=48"
                },
                "displayName": "M V",
                "emailAddress": "matovidlo2@gmail.com",
                "key": "JIRAUSER10000",
                "name": "test",
                "self": "http://localhost:8001/rest/api/2/
                user?username=test",
                "timeZone": "Etc/UTC"
            },
            "content": "http://localhost:8001/secure/
            attachment/10000/IssueTypes.png",
            "created": "2021-01-10T16:31:16.189+0000",
            "filename": "IssueTypes.png",
            "id": "10000",
            "mimeType": "image/png",
            "self": "http://localhost:8001/rest/api/2/attachment/10000",
            "size": 12880,
            "thumbnail": "http://localhost:8001/secure/
            thumbnail/10000/_thumb_10000.png"
        }
    ],
    "comment": {
        "comments": [],
        "maxResults": 0,
        "startAt": 0,
        "total": 0
    },
    "components": [],
    "created": "2021-01-10T16:31:16.350+0000",
"creator": {
    "active": true,
    "avatarUrls": {
        "16x16": "https://www.gravatar.com/avatar/
        e3bc3332c1a7f7b69b60fc77769f5f06?d=mm&s=16",
        "24x24": "https://www.gravatar.com/avatar/
        e3bc3332c1a7f7b69b60fc77769f5f06?d=mm&s=24",
        "32x32": "https://www.gravatar.com/avatar/
        e3bc3332c1a7f7b69b60fc77769f5f06?d=mm&s=32",
        "48x48": "https://www.gravatar.com/avatar/
        e3bc3332c1a7f7b69b60fc77769f5f06?d=mm&s=48"
    },
    "displayName": "M V",
    "emailAddress": "matovidlo2@gmail.com",
    "key": "JIRAUSER10000",
    "name": "test",
    "self": "http://localhost:8001/rest/api/2/user?username=test",
    "timeZone": "Etc/UTC"
},
"customfield_10100": null,
"customfield_10104": null,
"customfield_10105": "0|hzzzzz:",
"description": "h2. This is your first task. \n{color:#707070}
Issues are the things you do in a project. In business projects,
issues are called tasks. {color}\nh4. Types of tasks
\n{color:#707070}A task can represent a document, a creative
asset, a purchase and even a person.{color}\n!
IssueTypes.png!\nh4. Details\n{color:#707070}
The 'Details' section above, provides the information you need,
such as priority and status, to help you track the progress
of your tasks.{color}\nNext: [Workflows and statuses|MAN-2]
\n\n----\n[Learn more |http://blogs.atlassian.com/2015/11/
make-jira-core-issues-work-for-your-business-team/]",
"duedate": null,
"environment": null,
"fixVersions": [],
"issuelinks": [],
"issuetype": {
    "avatarId": 10318,
    "description": "A task that needs to be done.",
    "iconUrl": "http://localhost:8001/secure/viewavatar
    ?size=xsmall&avatarId=10318&avatarType=issuetype",
    "id": "10000",
    "name": "Task",
    "self": "http://localhost:8001/rest/api/2/issuetype/10000",
    "subtask": false
},
"labels": [],
"lastViewed": "2021-01-10T16:31:24.637+0000",
"priority": {
    "iconUrl": "http://localhost:8001/images/icons/
    priorities/medium.svg",
    "id": "3",
    "name": "Medium",
    "self": "http://localhost:8001/rest/api/2/priority/3"
},
"progress": {
    "progress": 0,
    "total": 0
},
"project": {
    "avatarUrls": {
        "16x16": "http://localhost:8001/secure/projectavatar
        ?size=xsmall&avatarId=10324",
        "24x24": "http://localhost:8001/secure/projectavatar
        ?size=small&avatarId=10324",
        "32x32": "http://localhost:8001/secure/projectavatar
        ?size=medium&avatarId=10324",
        "48x48": "http://localhost:8001/secure/projectavatar
        ?avatarId=10324"
    },
    "id": "10000",
    "key": "MAN",
    "name": "Management",
    "projectTypeKey": "business",
    "self": "http://localhost:8001/rest/api/2/project/10000"
},
"reporter": {
    "active": true,
    "avatarUrls": {
        "16x16": "https://www.gravatar.com/avatar/
        e3bc3332c1a7f7b69b60fc77769f5f06?d=mm&s=16",
        "24x24": "https://www.gravatar.com/avatar/
        e3bc3332c1a7f7b69b60fc77769f5f06?d=mm&s=24",
        "32x32": "https://www.gravatar.com/avatar/
        e3bc3332c1a7f7b69b60fc77769f5f06?d=mm&s=32",
        "48x48": "https://www.gravatar.com/avatar/
        e3bc3332c1a7f7b69b60fc77769f5f06?d=mm&s=48"
    },
    "displayName": "M V",
    "emailAddress": "matovidlo2@gmail.com",
    "key": "JIRAUSER10000",
    "name": "test",
    "self": "http://localhost:8001/rest/api/2/
    user?username=test",
    "timeZone": "Etc/UTC"
```

```
        },
        "resolution": null,
        "resolutiondate": null,
        "status": {
            "description": "",
            "iconUrl": "http://localhost:8001/images/icons/
            status_generic.gif",
            "id": "10000",
            "name": "To Do",
            "self": "http://localhost:8001/rest/api/2/status/10000",
            "statusCategory": {
                "colorName": "blue-gray",
                "id": 2,
                "key": "new",
                "name": "To Do",
                "self": "http://localhost:8001/rest/api/2/
                statuscategory/2"
            }
        },
        "subtasks": [],
        "summary": "This is your first task",
        "timeestimate": null,
        "timeoriginalestimate": null,
        "timespent": null,
        "timetracking": {},
        "updated": "2021-01-10T16:30:16.188+0000",
        "versions": [],
        "votes": {
            "hasVoted": false,
            "self": "http://localhost:8001/rest/api/2/issue/
            MAN-1/votes",
            "votes": 0
        },
        "watches": {
            "isWatching": false,
            "self": "http://localhost:8001/rest/api/2/issue/
            MAN-1/watchers",
            "watchCount": 0
        },
        "worklog": {
            "maxResults": 20,
            "startAt": 0,
            "total": 0,
            "worklogs": []
        },
        "workratio": -1
    },
    "id": "10000",
    "key": "MAN-1",
    "self": "http://localhost:8001/rest/api/2/issue/10000"
}
```

r