DEPARTAMENTO DE ENGENHARIA
INFORMÁTICA

Deep Learning Web Services

Faro

2019/2020

1170638 Lia Nisa Monteiro Meireles

Instituto Superior de Engenharia do Porto

Deep Learning Web Services

Faro

2019 / 2020

1170638 Lia Nisa Monteiro Meireles

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

Licenciatura em Engenharia Informática

« julho 2020»

Orientador ISEP: **Jorge Coelho**

Supervisor Externo: **Hooshiar Zolfagharnasab**

«To my family and friends,

for believing in me and in my dreams.»

Abstract

Nowadays, the application of Deep Learning algorithms in software products is increasing. The accuracy, facility and help that it brings allows clients to easily use tools and applications.
The *Deep Learning Web Service* project consists in the development of two services, the *Recongnition Service* and the *Retraining Service*, that will allow the detection of shapes and text information contained in technical drawings. The services were implemented in Python, using Mask R-CNN and OCR models for the detection.
The gRPC framework was used to establish the communication between the *Technical Drawing Annotation Tool* and the services. Besides the models used in the *Recognition Service*, it was also implemented the BestFit algorithm from the CAM2 company software. The process flow consists in the upload of a technical drawing, in the *Technical Drawing Annotation Tool*, that will be sent to the *Recognition Service*. The service will apply the Mask R-CNN model to the drawing, which results in the creation of masks that represent the shapes and textboxes detected. The BestFit algorithm will then be applied to these masks, in order to get the coordinates for the shapes. The data detected will be sent back to the client in a JSON format, through a protocol buffer.
As for the *Retraining Service*, it allows the retraining of the Mask R-CNN model, receiving a JSON file from the client, with corrected shapes coordinates. This will allow the improvement of the accuracy of the model used.
The projects solution implements the main tasks predefined, fulfilling the initial goal. With the work developed, new concepts were learned and put into practice, which lead to a satisfactory internship.

Contents

List of Figures

List of Tables

Notation and Glossary

Anaconda	Free and open-source distribution of the Python and R programming languages for scientific computing
API	Application programming interface
CAM2	Software product of FARO
Csharp	General-purpose, multi-paradigm programming language
dotNet Core	Open-source, general-purpose development platform for building cross-platform apps
Framework	Platform for developing software applications
FURPS	Functionality, Usability, Reliability, Performance, Supportability
FURPS+	FURPS with the addition of design, implementation, interface and physic requirements
gRPC	Modern open source high performance RPC framework
HTTP	HyperText Transfer Protocol
JSON	Javascript Object Notation
Keras	Open-source neural-network library written in Python
Mask R-CNN	Method for object detection and instance segmentation
MVC	Model-View-Controller
OCR	Optical Character Recognition algorithm
PMI	Product and Manufacturing Information
Python	Interpreted, object-oriented, high-level programming language with dynamic semantics
REST	Representational State Transfer
Tensorflow	An end-to-end open source machine learning platform

1. Introduction

This initial chapter serves as an introduction to the context of the present project. It will contain a brief presentation of the problem, it's objectives and the approach that was taken. After that, it is shown the work's contributions and the project's planning in order to provide a temporal perspective of the elaborated activities.
At the end of this chapter is a summary that details the structure of the report.

1.1 Context

This project is part of the course unit Projeto/Estágio (PESTI) of the Bachelor Degree in Informatics Engineering (LEI) of Instituto Superior de Engenharia do Porto (ISEP) and was developed in an external organization in the form of an internship.
As the Bachelor comes to an end, the students have the need to apply the knowledge acquired into real world problems. With this course unit, PESTI represents the culmination or the academic path.
When confronted with the possibility to develop this project in a company, the desire to contact with a business environment achieves it's goal. It could be part of our future and having this step taken while finishing the degree, allows us to gain more experience and maturity, as we are faced with impacting problems and solutions.
This project's challenge is to develop a web service that exposes deep learning and other data driven AI concepts, having been accepted for the opportunity to add value to the company and immerse in new technologies and fields.
The internship was carried in the company Faro. Nowadays, this company is a source for 3D measurements, imaging and realization technology, producing high precision solutions for 3D capture and measurement and also develops Computer Aided Manufacturing (CAM) software.

1.2 Problem's Description

The Web Services AI project consist in, as the name tells, developing a web service that exposes deep learning and other data driven AI concepts to better assist users int the preparation of measurement files for the company's software.
In this work field, nowadays, there are many companies developing high precision software solutions for 3D measurements. These 3D measurements can also be imported

into the software as technical drawings (Computer-aided design files) and compared with points cloud obtained through the measurements made.
In order to facilitate the comparison addressed, the Research Team from Faro had the idea to implement Deep Learning algorithms in order to proceed with the recognition of the information presented in the technical drawings imported. This approach began to be developed during one of the Hackathons that the company holds every year.
The usual situation consisted in the clients having to import a technical drawing and comparing the data, which also turned out to include creating some objects that were not presented and adding the correct PMI (Product and Manufacturing Information). This task took an unwanted amount of time.

1.2.1 Objectives

One of the main keys a buyer looks for, when acquiring a specific product, is it's practicality and see if it meets the needs. If we study the problem referred before, we can acknowledge that the actual solution is not the most efficient, turning out to actually be time-consuming and in need of a high level of knowledge of the domain.
Taking in consideration the analysis made, the main objective of the solution proposed would be to make the process described more efficient with the aid of deep learning algorithms able to detect and understand shapes and technical annotations. Deconstructing this big objective into small tasks that need improvement, we have:

- Detection of technical drawings semantic annotations;
- Recognition of the figures presented in this drawings;
- Automatic association between the figures presented and the respective annotations;
- Auto-ballooning tool;

These tasks will make it possible to reduce the time it takes to correct a technical drawing, magnifying the precision rate of the drawing in comparison with the piece inspected.

1.2.2 Approach

In order to achieve the main goals described before, two connected internships were created. On one side, we have the *Deep Learning Web Service* project and on the other side, there is the *Technical Drawing Annotation Tool*. The first shall expose data driven AI concepts in order to assist the users of the tool developed in the second project mentioned. This tool will allow it's users to properly highlight and annotate a technical drawing in a way that it can be interpreted back to an inspection program by FARO's CAM2 Software.

The approach made for the *Deep Learning Web Service* consisted, primarily, in building a Web Service in Python, able to communicate and establish a connection with the Technical Drawing Annotation Tool. This Web Service should be implemented following the Microservices Architecture [Wasson, 2019]. In addition, this service should implement the Mask-RCNN [He et al., 2018] framework for object instance segmentation. In order to do so, the Research Team had already the model trained and the database generated, missing the adaptation and inclusion into the web service developed.

The last stage that would allow the achievement of the objectives previously presented consists in processing text information from the drawings provided. This technical information is called PMI gives data such as geometric dimensioning, tolerancing and text for 3D annotations [Siemens, 2015].

1.2.3 Contributions

The main contribution of this project is the efficiency and practicality achieved through the implementation of a web service capable of understanding and recognising the technical drawings sent by the client.

In terms of innovation, it will also be the first time that a company, in this work field, implements a software solution using deep learning algorithms in order to facilitate the work and time spent by a client in submitting and adding a new technical drawing.

Another contribution is the completion of a proof of concept that emerged during one of the Hackathons that the company holds. With the help and guidance of the Research Team and their previous implemented work, a foundation was developed, allowing the company to continue to research and improve the models used, in order to achieve, in the future, results more accurate results and the detection of more complex figures and symbols.

1.2.4 Work Planning

During the time that the project was developed, there were several meetings with the Research Team in order to define and discuss the steps and approach that should be implemented.
Since the company implements the Scrum framework [Drumond, 2017] and Agile methodology [Sharma et al., 2012], the team hold daily meetings and carried out every two weeks sprints reviews. The main tasks demonstrated in the table 1.1 describe the elements that were demonstrated and discussed in these sprint reviews.

Task	**Date**
Adaptation and welcoming sessions	02.02 – 07.02
gRPC Web Service implementation	08.02 – 20.02
Modification and implementation of the Mask-RCNN existent algorithm	21.02 – 05.03
Development of best fit algorithms in Python (rectangles, circles and roundslots)	06.03 – 12.03
Implementation of the BestFit algorithm from CAM2	13.03 – 26.03
Model corrections, manual tests and research about PMI	27.03 – 09.04
Finding the syntax of PMI and defining 2D rules for 2D graphics	10.04 – 23.04
Implementing PMI parser and writer	24.04 – 07.05
Defining structures to include corrected data and loading old JSON files, update the details of the corrected shapes	08.05 – 21.05
Implementing the model for text boxes detection	22.05 - 04.06
Testing the implemented backend to find bugs and correct them	05.06 - 12.06

Table 1.1: Work planning

The main objectives were achieved, although the work planning changed due to the COVID-19 pandemic and the adaptation to remote work. In addition, there were complications with the connection to the VPN and the company servers, postponing some of the tasks.
For the sprint reviews every two weeks, there was also the need to elaborate presentations containing the achievements and work developed. These presentations were brief, with a duration of 5 minutes per intern, being hold amid the afternoon of Thursdays.

1.3 Report structure

Apart from the introduction, this dissertation contains four more chapters.

In the second chapter is described the State of Art, addressing the project's work related and existing tools.

The third chapter, Solution's Analysis and Design, offers an overview of the engineering involved in the achievement of the actual solution.

As for the fourth chapter, the Solution's Implementation, it explains crucial details to take attention in the implementation of the project and it presents the tests performed.

The fifth and final chapter addresses the conclusions drawn from the internship and the project development process.

2. State of Art

This chapter starts with a brief reference to the requirements analysis since it is a step crunch to determine the quality of the requirements and to the development of the software.
The following section addresses the existing work related to the tool being developed and the approach taken by similar solutions.
The three following sections are based in the research made before the actual implementation. These sections describe the main API Protocols that are used and the benefits of each of them, the most known objects detection frameworks and the principal deep learning libraries.
From the technologies that will be discussed, each section contains one of the technologies that was chosen to be employed.
Finally, we conclude with the analysis to the existing work methodologies will conclude the chapter.

2.1 Requirement Analysis

The term Requirement Analysis refers to the process of defining user needs and expectations for the software being built [Paradigm, 2017]. It is through this task that the requirements are captured, analyzed and validated, being an important phase in the software's development. So as to accomplish a well defined requirements analysis, the interest parts must reunite to clarify the problem's domain and the restrictions that this might have. This review must be documented since it will impact the way that the intended software is developed.
The most known models used for the requirements analysis are RUP and FURPS+, becoming two crucial models to study.

2.1.1 Rational Unified Process (RUP)

The Rational Unified Process was created by one of IBM's divisions, the Rational Software Corporation. It was designed as a software engineering process able to provide a disciplined approach to the assignment of tasks and duties within the development team. The main goal of this process is to secure the production of a software with high-quality, ensuring also the needs of its end-users. Furthermore, not only the quality of the software matters, but also the respect for the budget and time defined [Rational, 2001].
All the demanding tasks are provided to every team member, having base guidelines to let all members access the same knowledge about the project. The ensuring of the same level of information within the team enhances team productivity, since the process and view of development is known by all the team members.
The RUP model also defines patterns for the use of the Unified Modeling Language (UML). This language was also developed by the Rational Software Corporation and is, nowadays, the standard for the software's modeling and specialization. Since it is globally used, it guarantees that different teams can communicate efficiently during the all process of the software development [Rational, 2001].
Therefore, the use of the model RUP allows the documentation of the different phases of development in solid models and built with a syntax that is globally known. However, this model is not applicable to all the software projects, being the best solution to adapt it to every organization or project needs.
This model also adopts a set of the best practices to take into attention in software development projects. From a range of five practices, we can better understand the importance of approaching software iteratively; the need to have elicit and organized required functionalities and constraints; the benefits of using component-based architectures, promoting software reuse; the use of UML to visually model software and see how elements of the system fit together; the benefit that comes from having quality assessment built into the process and the ability to manage changes and track the changes made, having the team together to work as a single unit [Rational, 2001].
The figure 2.1 describes the different phases of the software development process.

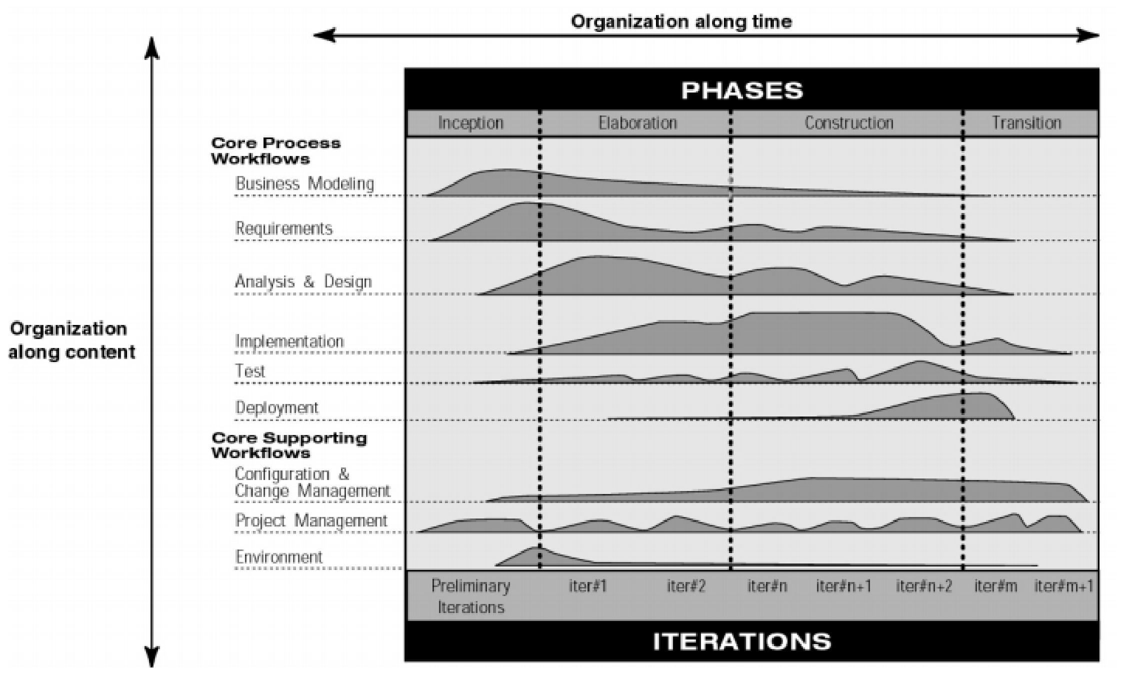


Figure 2.1: The Iterative Model graph.

Source: [Rational, 2001]

It's horizontal axis represents time and displays the dynamic aspect of the process through its execution, in terms of cycles or iterations. On the other axis, it is described in terms of activities, artifacts and workflows the static aspect of the process. From this image we can conclude that each core process lingers through iteration and phase, being that different projects or teams can have longer or shorter phases for each workflow.

2.1.2 FURPS+

The FURPS model was developed firstly by Grady at the Hewlett Packard company. It is a system for requirements classification of a software project. This model classifies the requirements following five categories: Functionality, Usability, Reliability, Performance and Supportability. The "+" of the FURPS+ acronym allows us to specify constraints, including design, implementation, interface, and physical constraints [Eeles, 2004].
From the category Functionality we have the functional requirements of the system, translating directly into functionalities of the software. The other ones are non-functional requirements, corresponding to technical conditions needed to satisfy the client's requests. One of the particularities of this classification is the emphasis put on understanding the different types of non-functional requirements. **Note:** Melhorar FURPS+

2.2 Related Work

A computer aided design (CAD) application can be used by a designer to create a drawing composed of a different number of components. The designer can also identify the assembly drawing components by attaching each of them to an identifier, which can be a letter, a number or label [Mikulecky, 2005]. In the figure 2.2, we see an example of these drawings, having represented it's technical information.

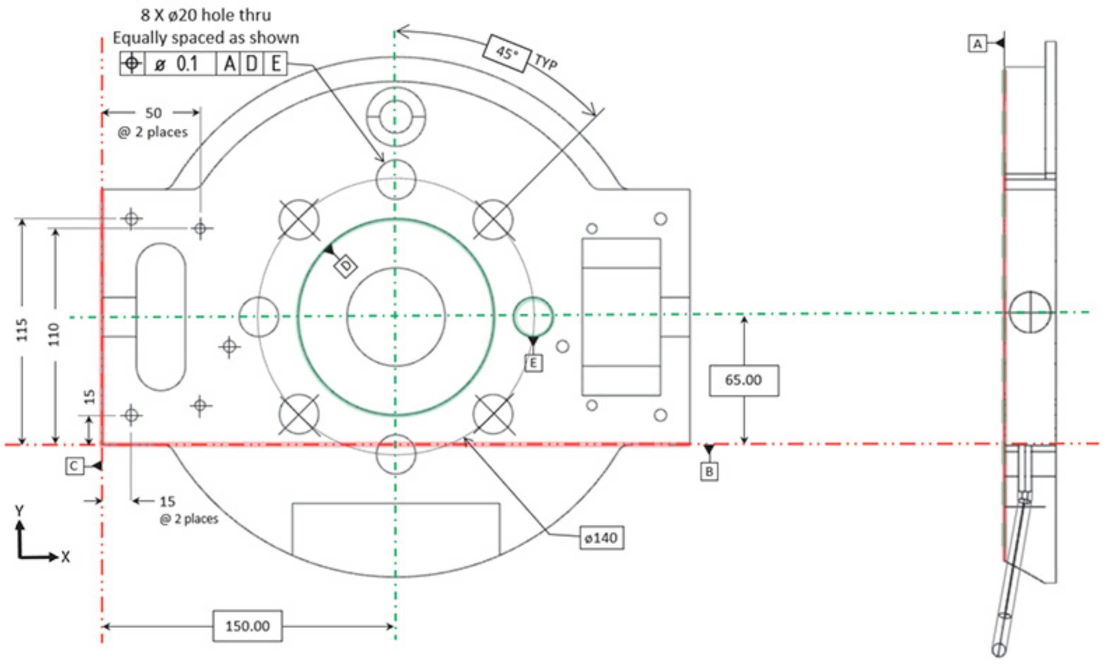


Figure 2.2: Technical drawing example.

Source: [Faro, 2018]

Each component of the drawing can be identified by attaching it to balloons. The balloons can be attached by lines referred to as "leaders". This process of identifying components of an assembly drawing by attaching the components to balloons by leaders is referred to as "ballooning" [Mikulecky, 2005].

The following software applications represent some of the present market offer for auto-ballooning applications, having functionalities similar to the ones being developed. Even though there are similarities, none of these applications makes use of deep learning algorithms in order to decrease the time taken by a client during the processing of a technical drawing.

2.2.1 InspectionXpert

The InspectionXpert Ballooning Software is a precision manufacturing software for ballooning First Article Inspection (FAI) Reports, AS9102 Compliance (standardizes qual-

ity management system requirements), and Production Part Approval Process (PPAP) [InspectionXpert, 2004].
Focusing on the ballooning functionalities that this tool provides, in general, it extracts data in one step. The following topics are an explanation of the possible uses that this tool has:

- Automatically read geometric dimensioning and tolerancing (GD&T) callouts and create a feature control frames. This is achieved in the application. selecting the text information of the draw being treated. After selecting, the software recognises the data and creates a table with its values;
- Add, remove, renumber and insert balloons. This makes possible to alter the existing data about the components;
- Compare and replace part revisions, then edit balloons to match;

For the work being developed, the most relevant part to take into consideration is the way that this software can detect the text information and withdraw its information, being able to localize it and identify the existing characters.

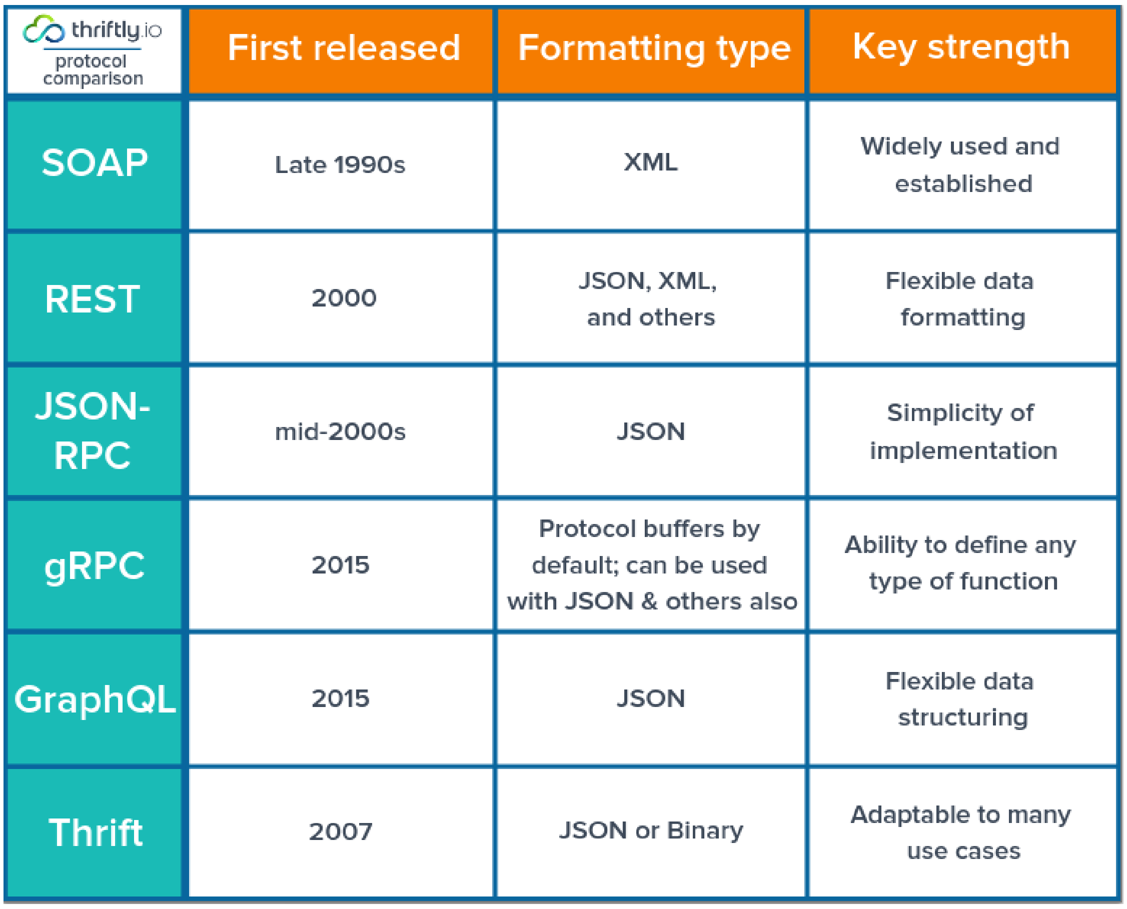
2.2.2 DISCUS

The DISCUS Desktop is an application able to manage technical data and identify characteristics and requirements from 2D drawings and specification documents. Combined with Google-based optical character recognition (OCR), the auto ballooning tool also allows the editing of the technical information and its recognition, creating sheets with the selected data [DISCUS, 2006].
Based on the information present in the company's website, one of the benefits of the DISCUS Desktop application is that it reduces the time taken to produce a First Article Inspection (FAI) and ballooned drawing by 50% or 80%. The efficiency and practicality achieved with this software is also one of the main objectives of the actual project.

To conclude this section, it is important to refer that nowadays there are more offers to take in consideration, as QA-CAD Software [Guthrie, 2003] and High QA [QA, 2009], since all applications have an auto-balloon tool, allowing the change and extraction of GD&T and Product Manufacturing Information (PMI). In terms of research, the knowledge about the way that these companies developed their solutions and knowing which technologies they are implementing allows us to develop a new solution, add value to the work achieved and try new algorithms for the data recognition.

2.3 API Protocols

For the development of a Web Service, it was crucial to look into the existing API protocols and compare them, analysing their benefits and deciding which would fit better in the current scenario. This way,The most known protocols such as SOAP, RESTand gRPC were examined for decision making. The figure 2.3 is a primer on the top six application programming interface (API) protocols of today, including the ones mentioned before.



thriftly.io protocol comparison	First released	Formatting type	Key strength
SOAP	Late 1990s	XML	Widely used and established
REST	2000	JSON, XML, and others	Flexible data formatting
JSON-RPC	mid-2000s	JSON	Simplicity of implementation
gRPC	2015	Protocol buffers by default; can be used with JSON & others also	Ability to define any type of function
GraphQL	2015	JSON	Flexible data structuring
Thrift	2007	JSON or Binary	Adaptable to many use cases

Figure 2.3: API protocols review.

Source: [Riley, 2019]

2.3.1 SOAP

Simple object access protocol (SOAP) was developed concurrently by IBM and Microsoft in order to facilitate client/server communication, using mainly the hypertext transfer protocol (HTTP). protocol to invoke remote objects. It is also an XML protocol, allowing the access to services, objects and servers in a platform independent manner [Koftikian, 2001].

The main goals of the SOAP protocol were to provide a standard object invocation protocol built on internet standards (using HTTP for the transport and XML for data encoding), create an extensible protocol and payload format that can evolve over time, all this without message box-caring, pipeline processing, objects model, objects by reference or objects activation. Since it does not require an object model, the protocol can be implemented in any language, as long as the client sends a valid SOAP request [Koftikian, 2001].

2.3.2 REST

The Representational State Transfer (REST) architecture was created with the objective to facilitate the web services implementation, since the services based on SOAP were more complex and obligated to a careful study of the W3C recommendations. This way, REST represents a stateless architecture [Vieira, 2015].
REST is lighter and less verbose, easier to interpret and to implement, having the advantage of getting the most of the HTTP cache which leads to a better performance. Althought, it is based on conventional text-based messaging (JSON, XML,CVS over HTTP) which are optimized of humans, but are not the ideal choices for internal service-to-service communication [Indrasiri, 2018]. The figure 2.4 represents, in a simplified way, how the REST architecture works.

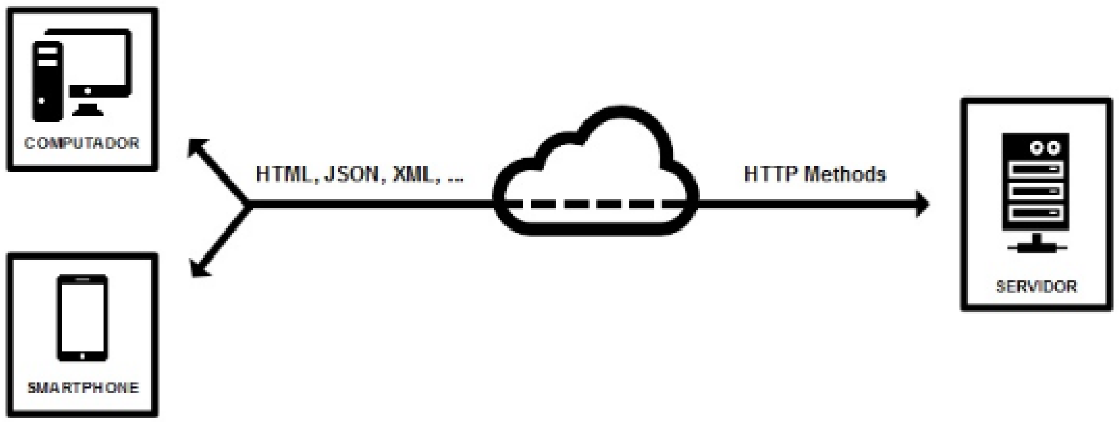


Figure 2.4: Simplified REST architecture.
Source: [Indrasiri, 2018]

Having an available service and knowing the uniform resource identifier (URI) of the intended resource, it can invoke that entity or functionality using the four methods provided by the HTTP protocol. Each resource is a representation that, usually, is built in JSON or XML, although any kind of response is valid if the web service indicates the MIME (Multipurpose Internet Mail Extensions) type of the response on the HTTP header [Indrasiri, 2018].

2.3.3 gRPC

The gRPC system was initially developed at Google in 2015. It is a high-performance open source and universal remote procedure call (RPC) framework, able to be run in every environment. It's documentation states that it can efficiently connect services in and across data centers. It uses HTTP/2 for transport and protocol buffers as the interface description language.
HTTP/2 primary goals is to reduce the latency by enabling full request and response multiplexing, minimize the protocol overhead with efficient compression of HTTP header fields and add support for request prioritization and server push. These characteristics will allow the applications to become faster, simpler and more robust [Grigorik, 2013].
A fundamental of gRPC is that it allows a customer to directly call methods on a server application of a different machine as if it were a local object, since the implementation of HTTP/2 and protocol buffers ensure the maximum interoperability. Since it is also a programming-language agnostic, we can use heterogeneous languages to build services and clients [Indrasiri, 2018].
If we compare the REST architecture and gRPC, for the implementation of a microservice, with REST, most of the external clients can consume the service as an API because most of them will know how to communicate with an HTTP RESTful service. Therefore, systems that require rapid iteration and standardized HTTP verbiage should be implemented in REST and systems that require a set amount of data or processing routinely, where the requester is either low power or resource-jealous should be implemented in gRPC [Sandoval, 2018].

2.4 Objects detection frameworks

Object recognition refers to the problem of detecting and localizing generic objects from different categories, such as cars or people, being a problem since objects from these categories can deeply vary in appearance [Zhao et al., 2010]. The object detection task generally consists of different subtasks due to the different categories to be detected. This problem is also able to provide valuable information for semantic understanding of images and videos, including image classification [P. F. Felzenszwalb and Ramanan, 2010].
The generic process of object detection aims at locating and classifying existing objects in an image and labeling them with rectangular bounding boxes, showing the confidence level of the classification made. In the figure 2.5, the difference of object detection and instance segmentation is visible, since instance segmentation is used in situations where we want to find the exact boundaries of our objects.

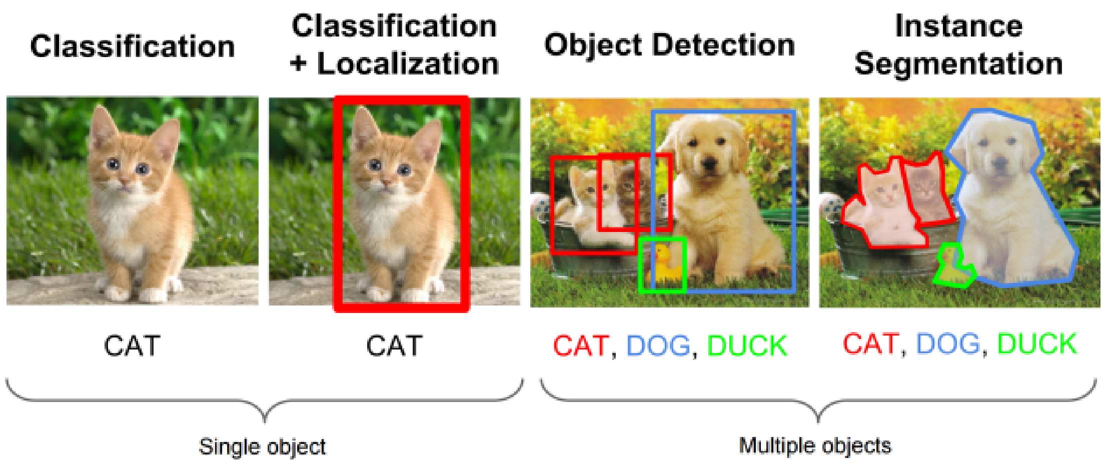


Figure 2.5: Difference between classification, localization, object detection and instance segmentation

Source: [Świeżewski, 2020]

The figure 2.6 demonstrates the two types of frameworks of generic object detection methods, having the first following traditional object detection pipeline (at first, it generates region proposals an then classifies each proposal into different object categories) and the second that regards the object detection problem as a regression or classification, adopting, for this, a unified framework to achieve the final results. The image also includes the methods from each type of framework and, from these, the ones that will be aboard are the R-CNN, the YOLO and the Mask-RCNN methods.

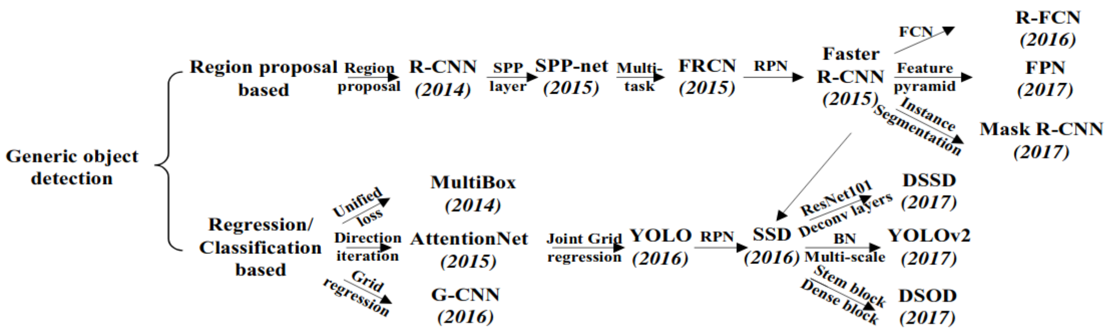


Figure 2.6: Two types of frameworks: region proposal based and regression/classification base

Source: [Zhao et al., 2010]

2.4.1 R-CNN

R-CNN stands for Regions with CNN features and was proposed by Ross Girshick [Girshick et al., 2014]. The main objective was to deal with the problem of efficient object canalization in the object detection problem.

To resolve it, it is used Selective search algorithm, taking advantage of segmentation of objects and Exhaustive search to efficiently determine the region proposals. With the Selective search algorithm,there are approximately 2000 region proposals. After, these regions are fed into a convolutional neural network that produces a 4096-dimensional feature vector as output. The CNN will then act as a feature extractor and the output dense layer consists of the features extracted from the image. Then, these features are fed into an support vector machine [Lorena and Carvalhor, 2002] in order to classify the presence of the object within that candidate region proposal.

The R-CNN algorithm, besides predicting the presence of an object within the region proposals, also predicts four values that are offset values to increase the precision of the bounding box. However, this algorithm takes a huge amount of time to train the networks, since there would be the need to classify 2000 region proposals and, since it uses a Selective search algorithm, which is fixed, it could also lead to the generation of worse candidate region proposals [Gandhi, 2018].The figure 2.7 shows a resume of the steps described and how the R-CNN proceeds.

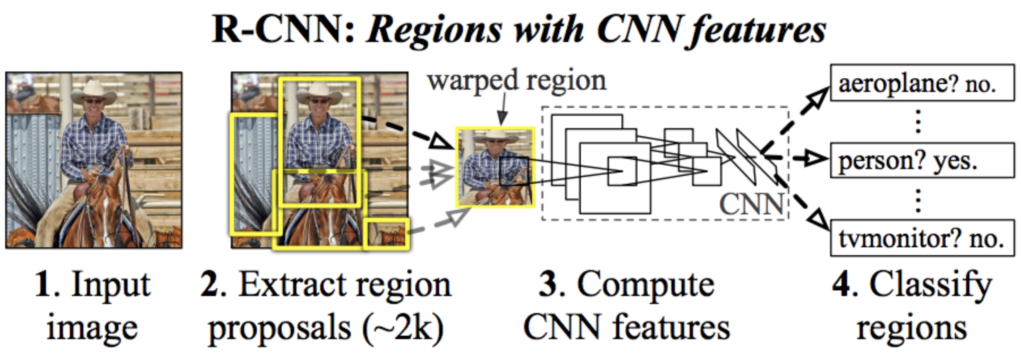


Figure 2.7: R-CNN: Main steps towards object detection with region proposals
Source: [Girshick et al., 2014]

2.4.2 YOLO

YOLO ("You Only Look Once") is an effective real-time object recognition algorithm that different from the region based algorithms.

The YOLO algorithm splits the an image into cells and each one of those cells is responsible for predicting mutiple bounding boxes and confidence intervals, since there can be more than one object in that cell. Each grid cell also predicts C conditional class probabilities which is used to encode both the probability of a determined class appearing in the box and verify how well the predicted box fits the object.The figure 2.8 represents the YOLO detection system. The mutiple boundind boxes and class probabilities for

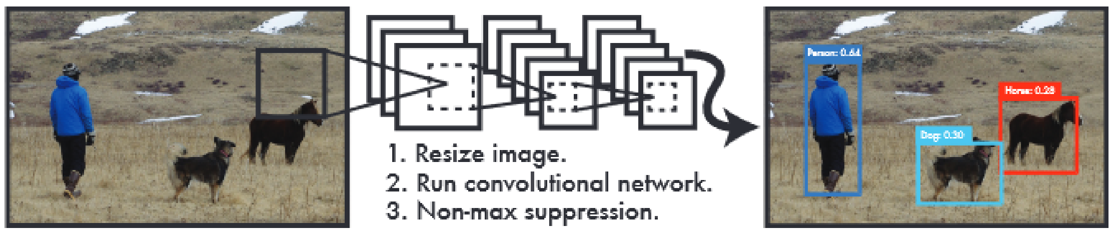


Figure 2.8: The YOLO Detection System.

Source: [Redmon et al., 2015]

those boxes are predicted simultaneously with a single convolutional network, turning the model extremely fast in comparison with traditional methods. One of the other advantages of YOLO is that it is highly generalizable, being less likely to break down when applied to new domains or unexpected input [Redmon et al., 2015].
Comparing YOLO and R-CNN, they do share some similarities since both use convolutional features for the proposed cells. However, YOLO puts spatial constraints on the grid cell proposals, helping mitigate multiple detections for the same object. It also proposes fewer bounding boxes compared to the 2000 from Selective search (R-CNN), turning this model more optimized [Redmon et al., 2015].
The YOLO algorithm was the first implementation choice of the Research Team due to these benefits. However, in a further stage, the Research Team opted for the Mask R-CNN algorithm.

2.4.3 Mask R-CNN

Mask R-CNN is described as a simple, flexible and general framework for object instance segmentation [RHe et al., 2018]. In general, it detects a objects simultaneously generating high quality masks for each instance.
This framework adopts two-stage procedure. First, the Region Proposal Network (RPN) proposes candidate object bounding boxes. The second stage, besides predicting the class and box offset, also outputs a binary mask for each detected object. The figure 2.9 demonstrates the process of the Mask R-CNN object instance detection.

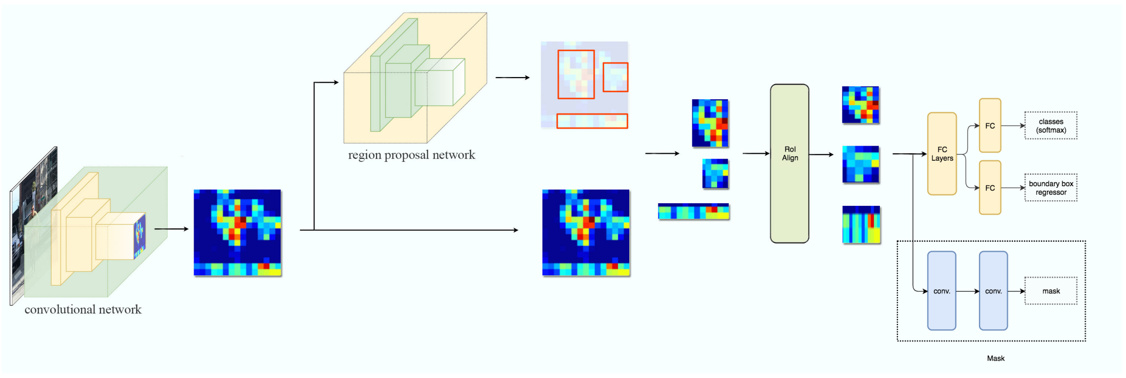


Figure 2.9: The Mask R-CNN process

Source: [Hui, 2018]

A mask encodes an input object's spatial layout. This extraction can be addressed naturally by the pixel-to-pixel correspondence provided by convolutions. The pixel-to-pixel behavior requires the Region of Interest (RoI) features to be well aligned to faithfully preserve the per-pixel spatial correspondence, motivating the development of the RoIAlign layer.

The RoIAlign layer proposes the avoidance of any quantization of the RoI boundaries or bins. It computes the value of each sampling point by bilinear interpolation [Jaderberg et al., 2015] from the nearby grid points on the feature map, leading to large improvements.

As mentioned before, the Mask R-CNN framework was the last approach of the Research Team, since the team verified that it achieve better results than the YOLO algorithm. The main objective of this project will be achieved using an implementation of Mask R-CNN, approached in the Implementation chapter.

2.5 Machine Learning libraries and platforms

Thousands of different programming languages have been created, and more are being created every year. In the Artificial Intelligence world and, as a subset, the Machine Learning area, Python is the most popular programming language, being the one preferred by the company in this field. The following subsections will address the libraries available to be used with Python and platforms that have been used over the years.

The platforms and libraries addressed are crucial for the implementation of MaskRCNN and the required data treatment to achieve the project objectives.

2.5.1 Tensorflow and Keras

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources, allowing

researchers and developers easily build and deploy machine learning powered applications.
It provides a collection of workflows to develop and train models using Python, JavaScript, or Swift, and to easily deploy in the cloud, in the browser, or on-device no matter what language you use.
Tensorflow has many valuable resources, such a:

- Models & datasets: pre-trained models built by Google and the community;
- Tools: ecosystem of tools to help its use;
- Libraries and extensioins built on it;

Running on top of TensorFlow, Keras was developed with a focus on enabling fast experimentation.It is an approchable, highly-productive interface for solving machine learning problems, with a focus on modern deep learning.

2.5.2 OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library, built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.
It possesses algorithms can be used to identify objects, segmentation and recognition, also having a statistical machine learning library and methods oriented to image processing.

2.5.3 COCO

COCO stands for Common Objects in Context and is a large-scale object detection, segmentation, and captioning dataset. It has several features like: object segmentation, recognition in context, 330K images, 1.5 million object instances, etc.
MaskRCNN model uses the the COCO dataset to build and train. This dataset can also be customized, allowing the creating of adaptations to existing models, feeding the networks with different images.

2.6 Work methodology

With the intuit to explain how the work was developed through the internship, the Scrum methodology was studied since it is the methodology proposed and implemented by the company. It will be explained in what this methodology consists and how the team shaped it to the team needs.

2.6.1 Scrum

Scrum is a process framework used to manage complex product development., making clear the relative efficacy of the product management and development practices, allowing the improvement of the team.

There are three pillars that uphold the implementation of empirical process control. First, we have Transparency that requires that the significant aspects of the process must be visible to those responsible for the outcome and it must be defined by a common standard so observers share a common understanding. Secondly, there is Inspection. The Scrum users must frequently inspect Scrum artifacts and progress toward a Sprint Goal, detecting the undesirable variances, However, this inspection must not get in the way of the work, being more beneficial when diligently performed by skilled inspectors at the point of work. Third, there is Adaptation. When the inspector notates that one aspect deviated outside the acceptable limits, the process must be adjusted as soon as possible, in order to minimize further deviation [Schwaber and Sutherland, 2013]. The figure 2.10 is a generic graphic on what the Scrum Process consists of.

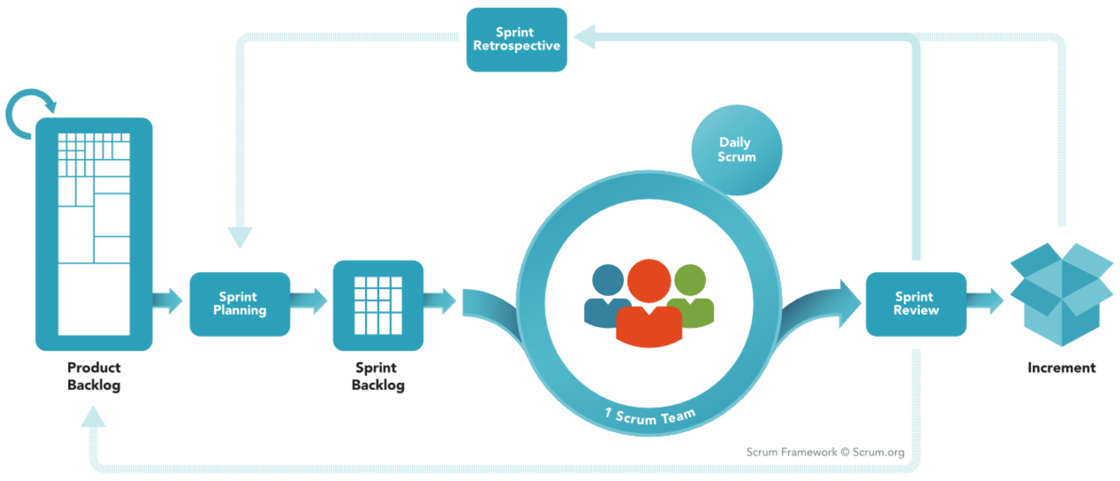


Figure 2.10: The Scrum Framework

The Scrum Team consists of a set of roles such as the Product Owner, the Development Team and a Scrum Master. The organization of the Research Team differs from the usual organization. It can be said that the members of this team are the Development Team, which consists of professionals who do the work of investigating and developing algorithms and methods that will better respond to the company needs. They develop also proof of concepts, supporting the other company teams developing the company's products.

As for the Product Owner, he is responsible for maximizing the value of the product and the work and the different Development Teams. In order to succeed, the entire organization must respect his or her decisions [Schwaber and Sutherland, 2013]. As for the Scrum Master that is responsible for ensuring that the Scrum Team adheres to Scrum theory, practices and rules, this is not a specific defined role in the Research Team.

The team also holds Scrum Events to create regularity and to minimize the need for meetings not defined in Scrum, The Sprint has a duration of two weeks and consists in the Sprint Planning, Daily Scrums, the development work and the Sprint Review. The Sprint Planning consists in defining the work to be performed in the Sprint, being created by the collaborative work of the entire Scrum Team. As for the Daily Scrum, everyday, for fifteen minutes, the team reunites to synchronize activities, create a plan for the next 24 hours and inspect the work done since the last Daily Scrum. As for the Sprint Review, it is held at the end of the Sprint to inspect the increment and adapt the Product Backlog if needed. During this review, all the company teams join for a presentation of the main objectives of the sprint, the work developed and the tasks achieved, as well as the problems encountered. This allows that different teams have a knowledge of what everyone is working on.

3. Solution's Analysis and Design

In this chapter we overview the engineering process adjacent to the implementation of the defined solution. It will expose the problem's domain, the functional and non-functional requirements analysis process and the solution's design.

3.1 Domain Concepts

The proof of concept chosen to validate the project's assumptions was the development of two microservices that will be used by the *Technical Drawing Annotation Tool*, a Windows Presentation Foundation (WPF) application. Both internship projects use well-defined concepts and objects for the recognition of technical drawings.
The shapes that may appear in the drawings are: rectangle, circle and roundslot. The figure 3.1 illustrates them.

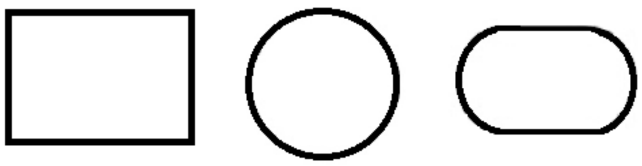


Figure 3.1: Shapes: rectangle, circle and roundslot

The rectangle is a four-sided flat shape with straight sides where all interior angles are right angles and the opposite sides are parallel and of equal length, This means that a square can also be a rectangle, and will both be treated as rectangles.
The circle is a round plane figure whose boundary consists of points equidistant from a fixed point (the centre point).It may have distinct radius but the centre point must be always equidistant from the boundary.
The roundslot is a shape that consists in a joint of two circles and a rectangle. The two circles must have the same radius and be aligned. The rectangle is formed based on the two circles, since two of it's sides are defined by the centre point of the circles. The circles must also be side-by-side, defining the semi-circle sides of the roundslot.
In addition to shapes, the drawings may also contain Product Manufacturing Information (PMI) data. The PMI can include information about geometric dimensions and tolerances, weld symbols, metadata and other definitive digital data. The figure 3.2 shows a

part with PMI annotations.

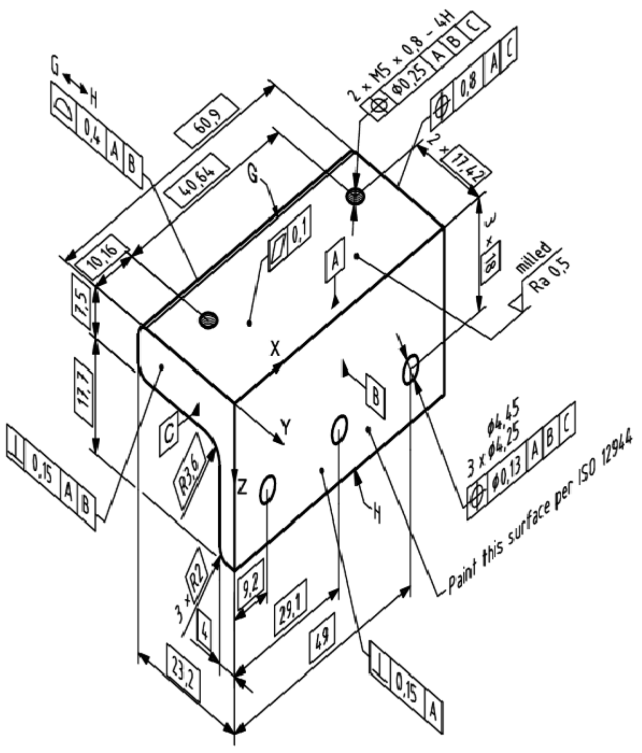


Figure 3.2: PMI annotations example

For the implementation of both microservices, there was no need of specifying a domain model. It does not implement neither defines the covered objects, but it's concept will be relevant for the solution design.

3.2 Non-functional Requirements

In the FURPS classification there are non-functional requirements, including the usability, reliability, performance and supportability. The successor of the FURPS model, FURPS+, comes up with the "+" signal, used to represent the limitations imposed by the system, allowing the specification of constraints like: design constraints, implementation constraints, interface constraints and physical constraints.
The non-functional requirements have maximum importance, they impose the resilience, responsiveness, elasticity and message driven characteristics described by the Reactive Manifesto.
The following sub-chapters will analyze each of non-functional requirements of the project.

3.2.1 Usability

The usability captures and states requirements based on user interface issues, for example, accessibility, interface aesthetics and consistency within the user interface. In the project, since there is not an interface with the user, the only non-functional requirement identified related to usability is:

1. The server side must be responsive and problems must be detected and corrected precociously, in a transparent and graceful way for the user

3.2.2 Reliability

The reliability constraints include aspects such as availability, accuracy and recoverability, for example, computations or recoverability of the system from a shut-down failure. In the project's context, there are different reliability requirements present, such as:

1. The server must be always available and working after failures.
2. The server must be resilient and capable of implementing new component instances after the failure of a component, guaranteeing high availability.
3. A component's failure must not propagate to other components.
4. The data related to each microservice must be contained in itself, since a failure that origins the data loss of a microservice must not be propagated to the other microservice, since there is no need to sync or share data.

3.2.3 Performance

This category includes restrictions involving system response time, recovery time and startup time. Since the project is based in microservices, these requirements are a fundamental factor:

1. The client requests, in the *Technical Drawing Annotation Tool*, must wait for the termination of the previous request, since it is not a multi client server service.
2. The system must stay responsive to different loads of information, having an elastic behavior.

3.2.4 Supportability

The supportability section includes the specification of the testability, adaptability, maintainability, compatibility, configurability, installability, scalability, localizability and others. For this project development, there were captured different requirements related with the supportability, such as:

1. The microservices must be open to expansion, allowing the implementation of new components or change of algorithms.

2. The service must be horizontally scalable.

3.2.5 Implementation Constraints

This type of restrictions affects how the system is build, putting limits on coding or construction, for example, required standards, platforms or implementation language. The following restrictions were captured:

1. The microservices must be implemented in Python and the clients implemented in .Net Core, in the Technical Drawing Annotation Tool

2. The Deep Learning algorithms to be used are MaskRCNN and OCR, already trained and implemented previously.

3. Due to the use of MaskRCNN, the Python version must be 3.5

4. The BestFit algorithm from *CAM2* project (company's product) must be called and used.

5. Must perform functional tests.

3.2.6 Interface constraints

Interface constraints affect the way how the communication between components is made. It refers to the description of protocols or the nature of the information that is passed across the interface. The constraints identified are:

1. The microservices must allow the communication with the Technical Drawing Annotation Tool.

2. The communication must be done through protocol buffers.

3.3 Functional Requirements

In the FURPS+ acronym, the F represents all the system functional requirements that would expect to see described. These define the main project features and can be architecturally significant operations.
In the project, these requirements are online related with the microservices functionalities. Its representation its made in the form of use cases.

3.3.1 Recognition Service

The Recognition Service aims detecting the shapes and text information of a received technical drawing image received from the client. The detected information must be sent back to the client. Since this use case is complex to explain and achieve, it was divided into several more simple to understand:

1. The client, through the Technical Drawing Annotation Tool, must be able to send a technical drawing to the Recognition Service.
2. The Recognition Service must detect the shapes drawn in the technical drawing, through the MaskRCNN algorithm.
3. The Recognition Service must create masks for the shapes detected withdraw the shapes countour points.
4. The Recognition Service must write the countour points information in a JSON file.
5. The Recognition Service must get the coordinates for the shapes, using the *CAM2* BestFit algorithm.
6. The masks information and shapes coordinates must be written to a JSON file.
7. The Recognition Service must create masks for the text boxes detected in the technical drawing received, through the MaskRCNN algorithm.
8. The Recognition service must get the countour points for the text boxes and write the information in a JSON file.
9. The Recognition Service must get the text boxes coordinates, from the *CAM2* BestFit algorithm.
10. The Recognition Service must remove the text boxes from the technical drawing and send them to the OCR algorithm, to detect its text information, writing it in a JSON file.
11. The shapes JSON file and the text JSON file must be send back to the client.

3.3.2 Retraining Service

The Retraining Service aims at allowing the retraining of the MaskRCNN model with corrected shapes information. In the Technical Drawing Annotation Tool, the clients will be able to resize, rotate and move the shapes previously detected. This tool will allow the clients to correct the information that the Recognition Service sent them. With the correct information being sent back to the Retraining Service, it will feed a new stage of training of the implemented model. Therefore, the use cases above will allow the construction of a new training phase:

1. The client, through the Technical Drawing Annotation Tool, must be able to send a JSON file with the corrected shapes information to the Retraining Service.

2. The Retraining Service must save the technical drawing and append to an existing JSON file the information about the corrected shapes.

3.4 Design

This section refers to the business adjacent concepts. It addresses the Deployment Diagram of the solution and a detailed description of the most important use cases to implement.

3.4.1 Deployment Diagram

The implemented solution involves the development of two microservices, Recognition Service and Retraining Service, using the gRPC framework. These microservices will be used by the *Technical Drawing Annotation Tool*, communicating through protocol buffers. The figure 3.3 demonstrates the solution's deployment diagram.

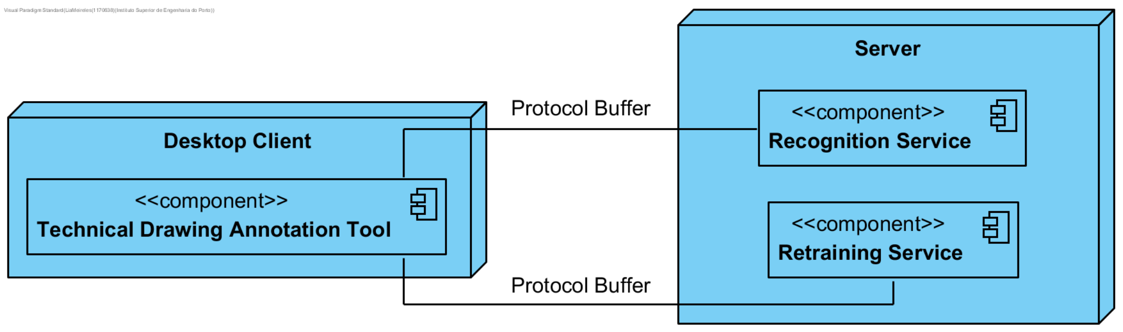


Figure 3.3: Solution's deployment diagram

Although the services can operate separately, for the Retraining Service to be able to save the new information from a corrected technical drawing, the Recognition Service must be obligatory used, since this is the service that will allow the client to apply changes in the detected shapes.

3.4.2 Recognition Service

The Recognition Service serves the purpose of detecting both shapes and text information that a received technical drawing contains. In the analysis, this use case was divided for a better understanding of the flow. The figure 3.4 shows the Use Case Diagram for the recognition of the shapes detected for the image received.This diagram was divided in two, being the first part related to the shapes' recognition.

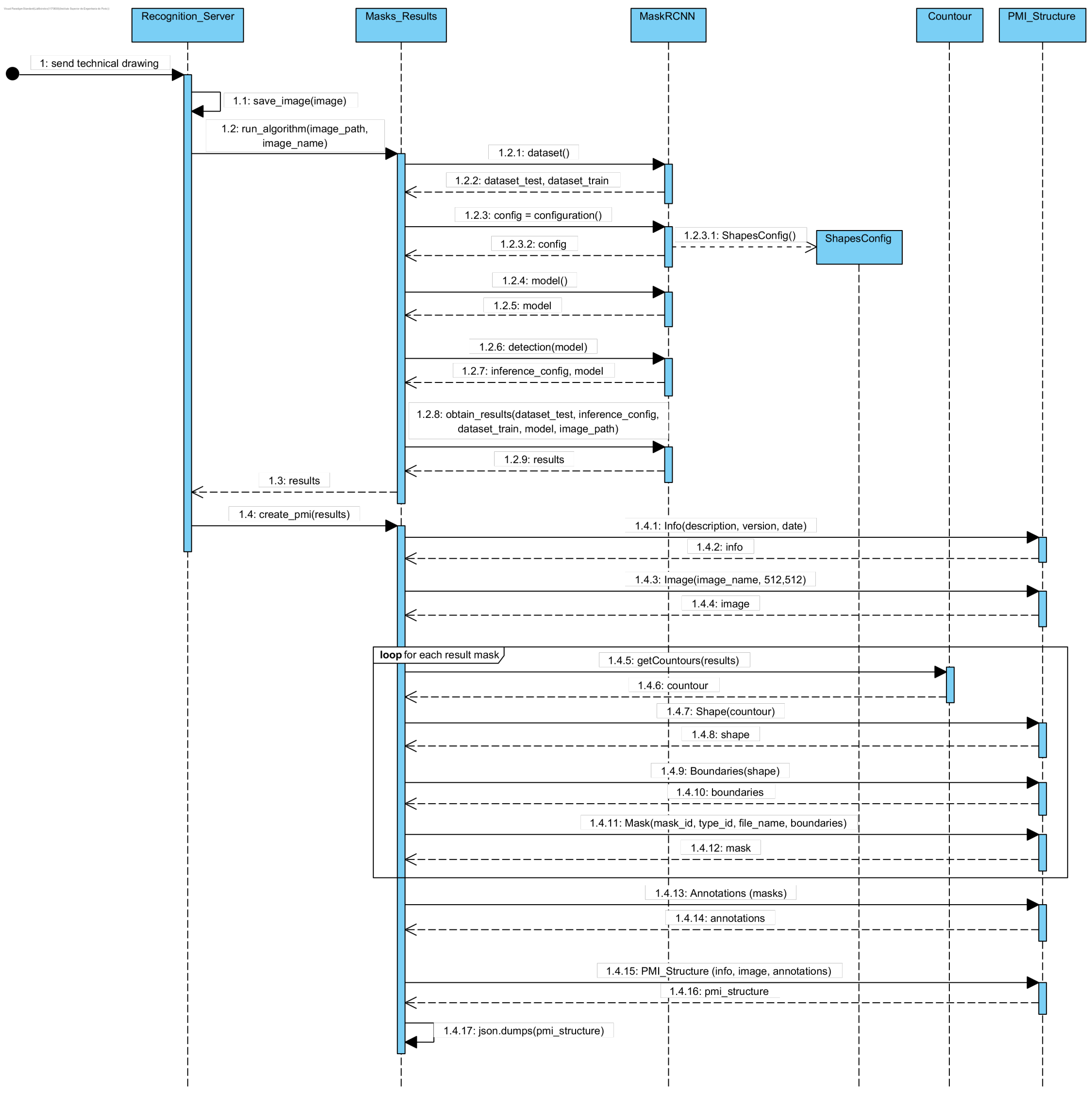

Recognition_Server
Masks_Results
MaskRCNN
Countour
PMI_Structure
1: send technical drawing
1.1: save_image(image)
1.2: run_algorithm(image_path, image_name)
1.2.1: dataset()
1.2.2: dataset_test, dataset_train
1.2.3: config = configuration()
1.2.3.1: ShapesConfig()
ShapesConfig
1.2.3.2: config
1.2.4: model()
1.2.5: model
1.2.6: detection(model)
1.2.7: inference_config, model
1.2.8: obtain_results(dataset_test, inference_config, dataset_train, model, image_path)
1.2.9: results
1.3: results
1.4: create_pmi(results)
1.4.1: Info(description, version, date)
1.4.2: info
1.4.3: Image(image_name, 512,512)
1.4.4: image
loop for each result mask
1.4.5: getCountours(results)
1.4.6: countour
1.4.7: Shape(countour)
1.4.8: shape
1.4.9: Boundaries(shape)
1.4.10: boundaries
1.4.11: Mask(mask_id, type_id, file_name, boundaries)
1.4.12: mask
1.4.13: Annotations (masks)
1.4.14: annotations
1.4.15: PMI_Structure (info, image, annotations)
1.4.16: pmi_structure
1.4.17: json.dumps(pmi_structure)

Figure 3.4: Sequence Diagram for Shapes Recognition

When the protocol buffer is received, it contains an image as a request. This image will first be processed by the *Masks_Results* for the shapes detection. The *Masks_Results* will load the model, *MaskRCNN*. The *MaskRCNN* will create the dataset for training and the dataset for testing, loading the configurations predefined. With it, it will load the model that will be applied to the technical drawing given. As a result from the detection, we get an array that contains the results, being also generated masks for each of the shapes detected. The results array contains information about the masks generated, such as *type_id*, *mask_id*, *mask_name* and *image_size*.

The *type_id* gives information about which shape was detected, being that each number is related with a different shape. These ids were defined previously, when the Research Team developed their own MaskRCNN model:

- Rectangle : 1
- Circle: 2
- RoundSlot: 3

After having the results and the masks generated, the JSON file that will contain the information about the shapes detected will be built. In the *create_pmi* method, the *PMI_Sructure* will be instantiated. It contains the definition for the many objects that will be present in the JSON file. The varied information that will be contained is:

- Information: description, version and date
- Image: id, file name, width and height
- Annotations: masks information

For each of the masks detected, it will be calculated the countour points and applied the BestFit algorithm, based in which shape was detected. The BestFit algorithm will use the information from the countour points in order to formulate the coordinates that best adjust to it.
After creating the JSON file with the shapes detected information, it proceeds for the textboxes and text recognition. The figure 3.5 represents the continuation of the use case diagram presented before.
Similar to the previous diagram, the *MaskRCNN* is used, but ony to detect the text boxes (rectangles) from the technical drawing. Having the masks reproduced by the algorithm, the countours are calculated in order to apply the BestFit algorithm to these regions. With the coordinates for the fitted textboxes, these will be, one by one, withdraw from the drawing and the OCR model will be applied to it. From the OCR model is achieved the text recognition. The JSON file will then contain the information about the textbox coordinates and the text that was recognised inside it. The figure 3.6 is an example of a part of a JSON file.

Having the JSON with the shapes detected and another JSON with the textboxes and text information detected, both will be sent back to the client, to the *Technical Drawing Annotation Tool*, that will visually reproduce the information received.

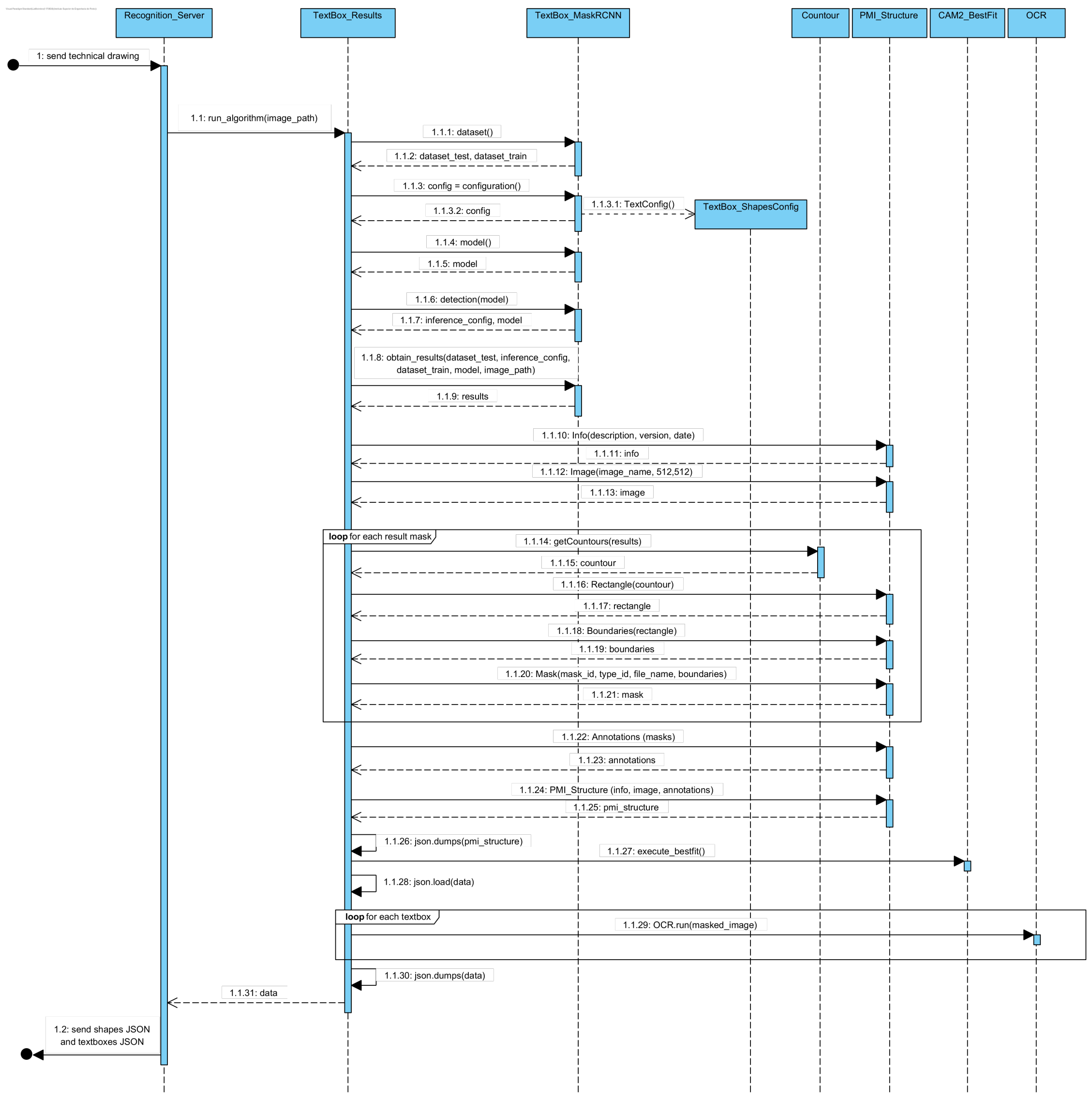
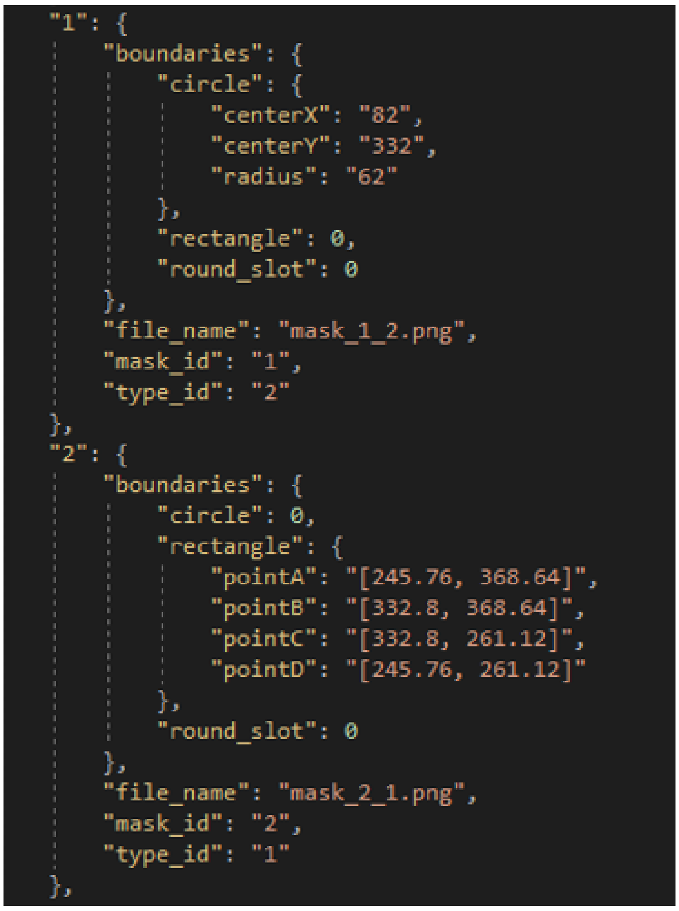

Recognition_Server
TextBox_Results
TextBox_MaskRCNN
Countour
PMI_Structure
CAM2_BestFit
OCR
TextBox_ShapesConfig
1: send technical drawing
1.1: run_algorithm(image_path)
1.1.1: dataset()
1.1.2: dataset_test, dataset_train
1.1.3: config = configuration()
1.1.3.1: TextConfig()
1.1.3.2: config
1.1.4: model()
1.1.5: model
1.1.6: detection(model)
1.1.7: inference_config, model
1.1.8: obtain_results(dataset_test, inference_config, dataset_train, model, image_path)
1.1.9: results
1.1.10: Info(description, version, date)
1.1.11: info
1.1.12: Image(image_name, 512,512)
1.1.13: image
loop for each result mask
1.1.14: getCountours(results)
1.1.15: countour
1.1.16: Rectangle(countour)
1.1.17: rectangle
1.1.18: Boundaries(rectangle)
1.1.19: boundaries
1.1.20: Mask(mask_id, type_id, file_name, boundaries)
1.1.21: mask
1.1.22: Annotations (masks)
1.1.23: annotations
1.1.24: PMI_Structure (info, image, annotations)
1.1.25: pmi_structure
1.1.26: json.dumps(pmi_structure)
1.1.27: execute_bestfit()
1.1.28: json.load(data)
loop for each textbox
1.1.29: OCR.run(masked_image)
1.1.30: json.dumps(data)
1.1.31: data
1.2: send shapes JSON and textboxes JSON

Figure 3.5: Sequence Diagram for TextBoxes Recognition



```json
"1": {
    "boundaries": {
        "circle": {
            "centerX": "82",
            "centerY": "332",
            "radius": "62"
        },
        "rectangle": 0,
        "round_slot": 0
    },
    "file_name": "mask_1_2.png",
    "mask_id": "1",
    "type_id": "2"
},
"2": {
    "boundaries": {
        "circle": 0,
        "rectangle": {
            "pointA": "[245.76, 368.64]",
            "pointB": "[332.8, 368.64]",
            "pointC": "[332.8, 261.12]",
            "pointD": "[245.76, 261.12]"
        },
        "round_slot": 0
    },
    "file_name": "mask_2_1.png",
    "mask_id": "2",
    "type_id": "1"
},
```

Figure 3.6: Example of a JSON file with the shapes data

3.4.3 Retraining Service

The use case of allowing the retraining of the MaskRCNN model consists in receiving, from the client, a JSON file which structure is the same as the previous JSON file sent, with the shapes detected information. On the *Technical Drawing Annotation Tool*, the user will have the chance to apply a set of operations to the shapes, such as resizing, rotating and moving. If the user desires to save the new information from the shapes that suffered corrections, the information will be saved in the JSON file received previously, overwriting the previous coordinates. Then, the user is able to send back to the Retraining Service the same JSON file, with the overwritten changes. The figure 3.7 demonstrates how the Retraining Service will proceed.

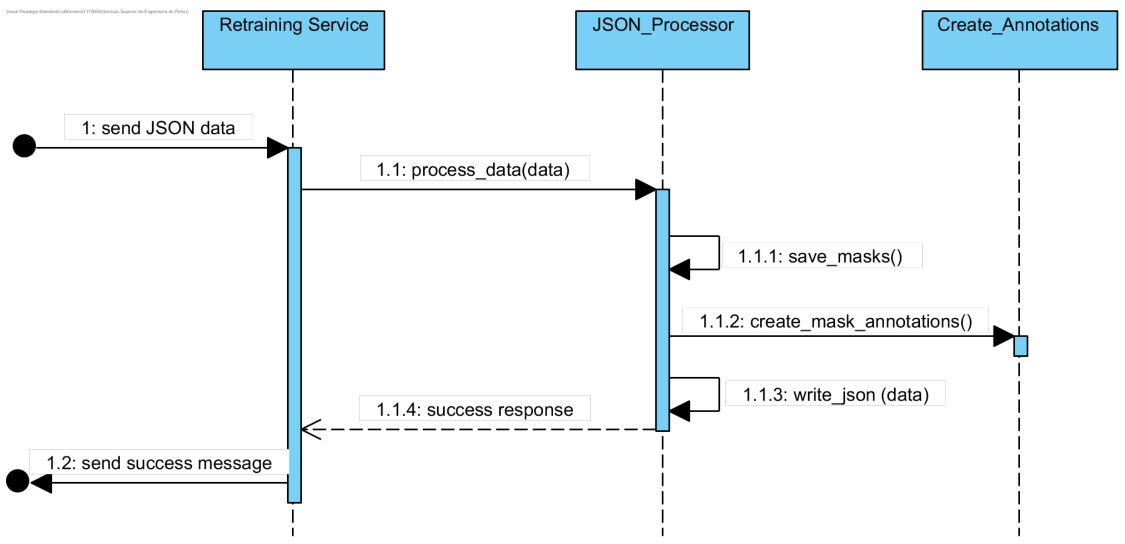


Figure 3.7: Sequence Diagram for Retraining Service Use Case

In the Retraining Service, the data received will suffer a structure change, in order for it to allow the retraining of the model. Having a specific JSON file that contains all the data from the initial train phase of the model, the received information will be included in this file. Since this file follows the COCO dataset format, which differs from our structure used previously, to send the shapes information to the *Technical Drawing Annotation Tool*, the main objective of this use case is to append the information to the COCO dataset file with the needed structure and requirements.

4. Implementation

In this chapter we present some highlights of the implementation and tests.

4.1 Implementation Description

In this section we present the result of the implementation of the final solutions. The first subsection purpose is to explain the implementation of the gRPC framework in the solution. The second subsection details how the MaskRCNN algorithm is used. These subsections are accompanied with code snippets taken from the solution implemented.

4.1.1 gRPC Implementation

With gRPC, the client application, the *Technical Drawing Annotation Tool*, can directly call a method on the server application, in this case the Recognition Service or Retraining Service on a different machine as if it were a local object, making it easier for you to create distributed applications and services. On the server side, the server implements this interface and runs a gRPC server to handle client calls. On the client side, the client has a stub that provides the same methods as the server. The figure 4.1 demonstrates a generic implementation of a server and client in two different languages.

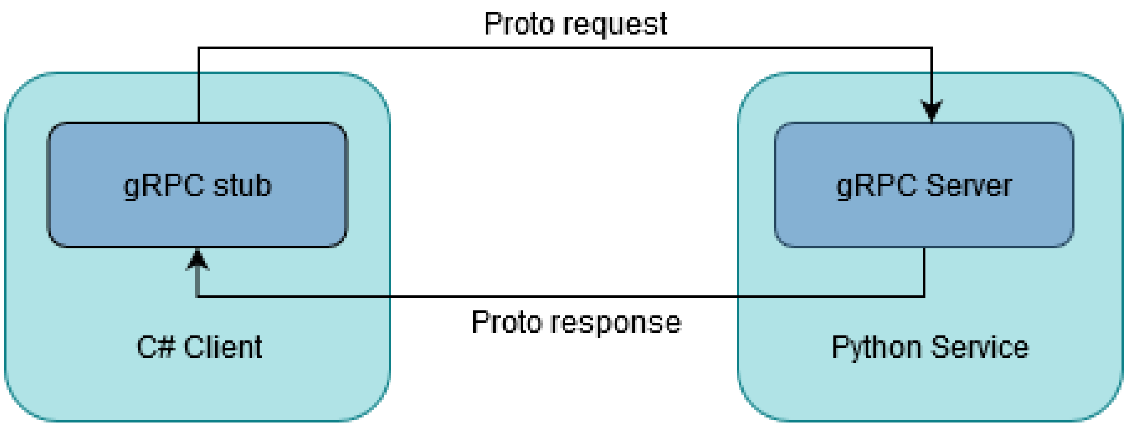
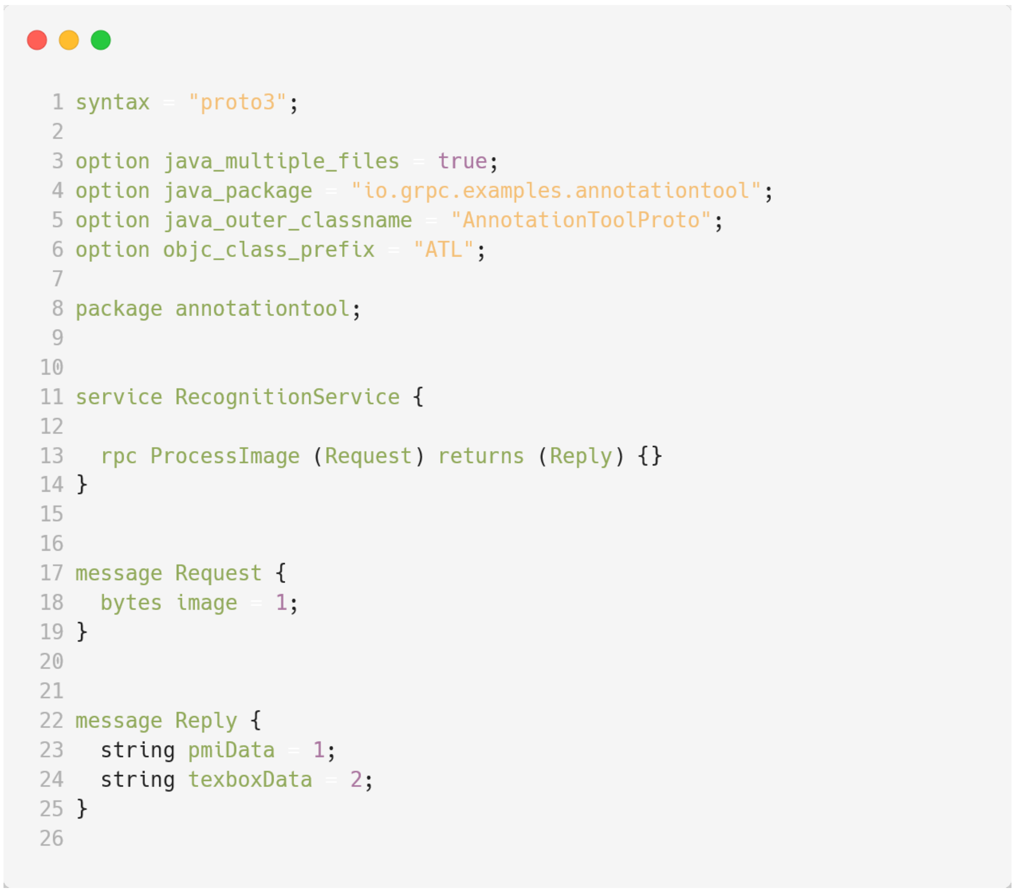


Figure 4.1: gRPC with a C# client and a Python service

Since gRPC uses Protocol Buffers, the first step when working with it is to define the structure for the data you want to serialize in a proto file. Having two services, two proto files were defined, one for the Recognition Service and another for the Retraining Service. The figures 4.2. and 4.3 are a snippet of these proto files.
In the figure 4.2, the Recognition Service has, as a message request, a vector of bytes,

that is the technical drawing to be sent from the client. The message reply contains two strings that represent each of the JSON data that will be sent, with the recognition of the shapes and textbox information.

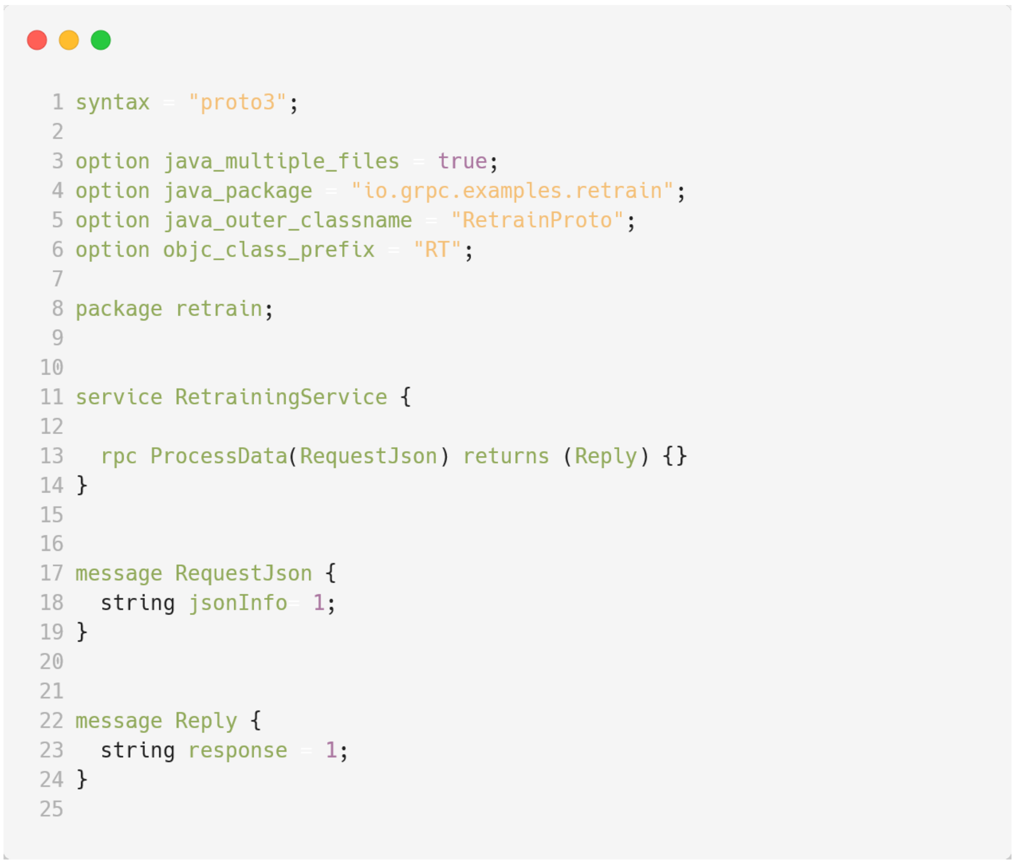


```
syntax = "proto3";

option java_multiple_files = true;
option java_package = "io.grpc.examples.annotationtool";
option java_outer_classname = "AnnotationToolProto";
option objc_class_prefix = "ATL";

package annotationtool;


service RecognitionService {

  rpc ProcessImage (Request) returns (Reply) {}
}


message Request {
  bytes image = 1;
}


message Reply {
  string pmiData = 1;
  string texboxData = 2;
}

```

Figure 4.2: Recognition Service proto file

In the figure 4.3, the Retraining Service expects a string that is the JSON information with the corrected information for the shapes detected and, as a response, it will send a string that declares the success or failure of the operation.

It was implemented an Unary RPC, where the client sends a single request and gets back a single response. This way, the server is always running, but is only capable of answering one request at a time.



```
syntax = "proto3";

option java_multiple_files = true;
option java_package = "io.grpc.examples.retrain";
option java_outer_classname = "RetrainProto";
option objc_class_prefix = "RT";

package retrain;


service RetrainingService {

  rpc ProcessData(RequestJson) returns (Reply) {}
}


message RequestJson {
  string jsonInfo= 1;
}


message Reply {
  string response = 1;
}

```

Figure 4.3: Retraining Service proto file

4.1.2 MaskRCNN algorithm

In this section we describe the implementation and use of the MaskRCNN by using a simple technical drawing as example. The figure 4.4 is a technical drawing composed with the three different shapes that the model detects: rectangle, circle and roundslot. Since the model being used is yet in a primary stage of training, the drawings are also simple.

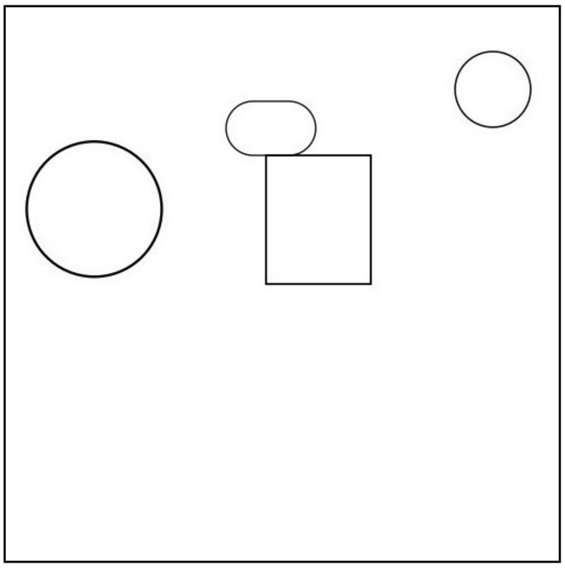


Figure 4.4: Example of a technical drawing

As described in section 3, the MaskRCNN model will be applied to the technical drawing. After running, it produces masks for the shapes detected. In this case, the masks produced are the following presented in figure 4.5.

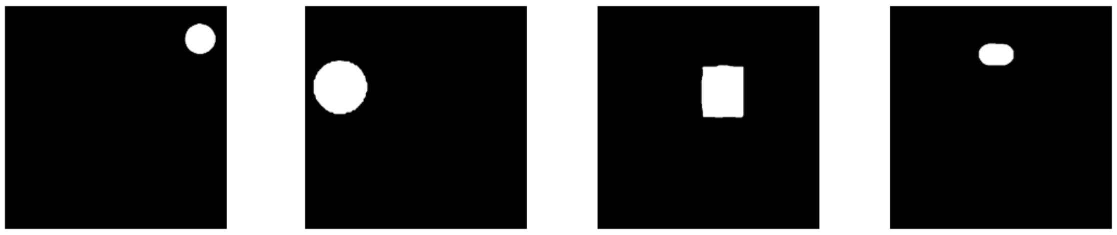


Figure 4.5: Masks produced by the MaskRCNN model

It is visible that the masks produced do not show perfect shapes. Since the model is yet not too accurate, there is a need to apply a BestFit algorithm that, based on these masks, will reproduce optimized shapes for the given countours. For this example, the masks obtained after applying the BestFit algorithm are presented in the figure 4.6.
It is notorious the difference between the masks in the figure 4.5 and the masks in the figure 4.6. The BestFit algorithm is implemented in the company's software product,

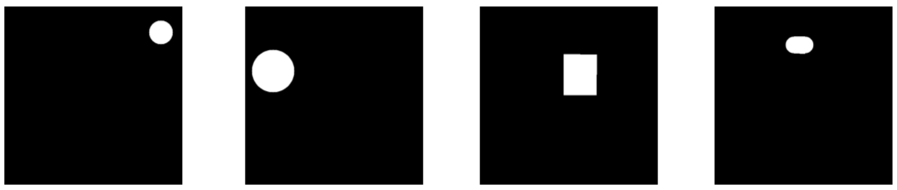


Figure 4.6: Masks after the BestFit algorithm

being called as an executable file. It produces shapes following it's mathematical theory. After having these masks, the shapes coordinates are written into a JSON file and passed to the client side.

4.1.3 JSON Structure

In order to increase the understanding of the JSON file used to write the information related to the shapes detection, the figure 4.7 breaks down the objects and its attributes that compose the JSON file. The major part of the information is obtained with the results produced by the MaskRCNN model.

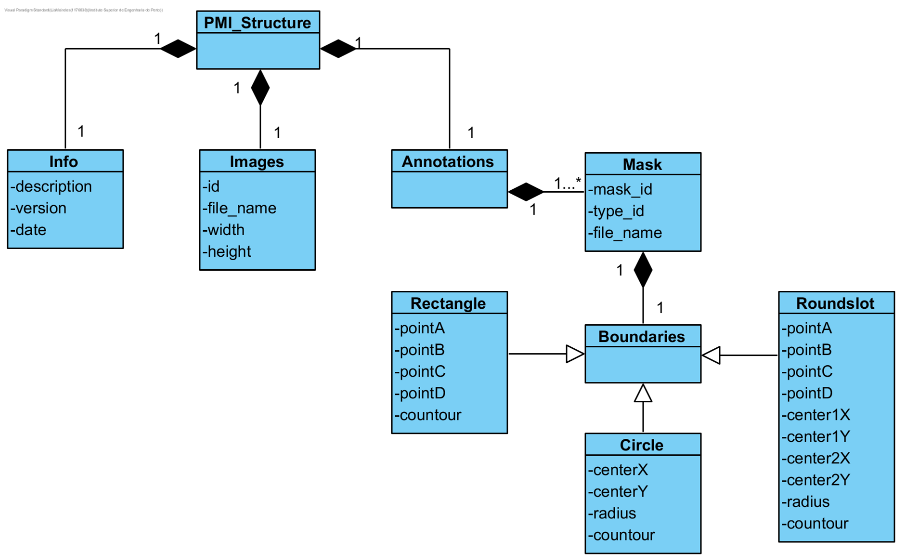


Figure 4.7: Class Diagram of the PMI Structure

4.2 Tests

We now describe the tests that are important to guarantee the quality of the solution. Once the domain objects are present in the *Technical Drawing Annotation Tool* project, the tests performed for this project consisted in Manual Tests and Acceptance Tests, using the tool developed in the other internship.

4.2.1 Manual Tests

Manual testing consists in executing test cases without using any automation tools. It is the most primitive of all testing types and helps find bugs in the software system.

Having a database with hundreds of trained technical drawing images, several tests were made, with drawings including text information and without text information. The image 4.8 shows the graphic interface of the *Technical Drawing Annotation Tool*, where the user inserts the desired drawing to be process. In this case, it is presented a technical drawing with PMI data to be detected.

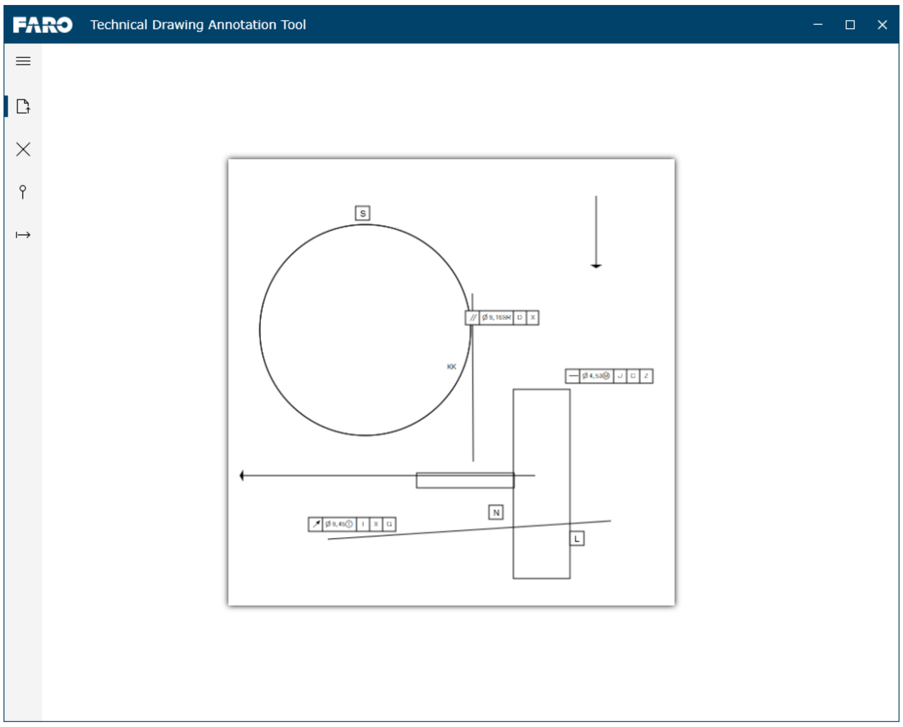


Figure 4.8: Graphic interface with a technical drawing example

After sending this image to the Recognition Service, the result achieved is presented in the figure 4.9. As it can be noticed, the shapes and textboxes were detected, although there is a deficiency in the text recognition. This deficiency is due to the application of the OCR algorithm that is yet little trained.

Another relevant manual test to show is the recognition that the server made before and after the implementation of the BestFit algorithm from *CAM2*. Since the work developed was incremental, before applying the algorithm, it was developed a best fit algorithm for each shape, in Python. In the figure 4.10, on the left, is the result of applying this BestFit algorithm and,on the right, is the result with the *CAM2* BestFit algorithm.

As it is notorious, the BestFit from *CAM2* software produces better-fitted shapes. With this, the Manual Tests were also important to understand the quality and performance of

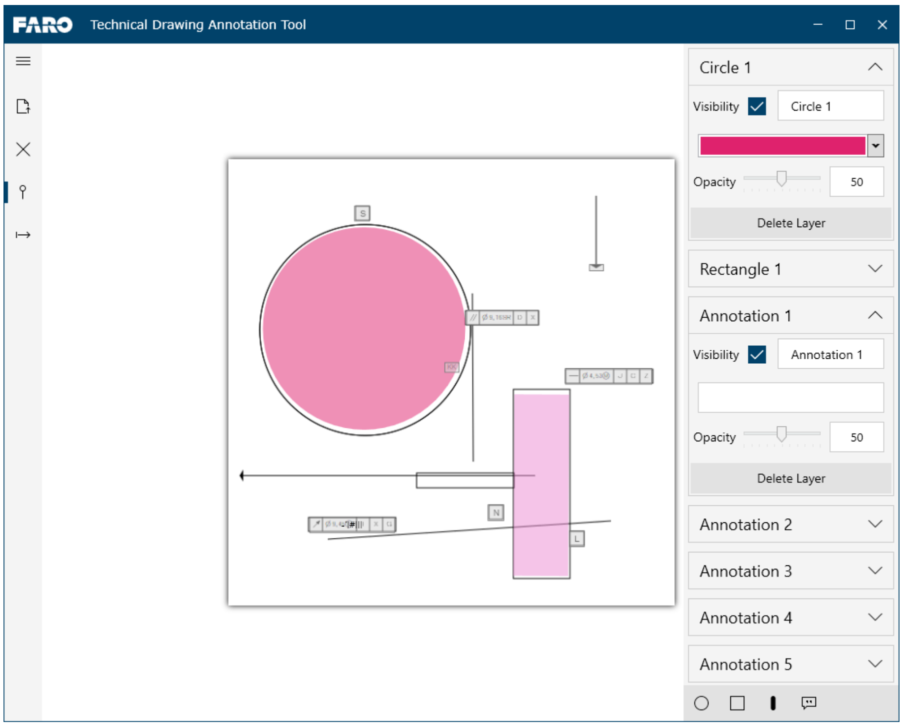


Figure 4.9: Graphic interface with a technical drawing result example

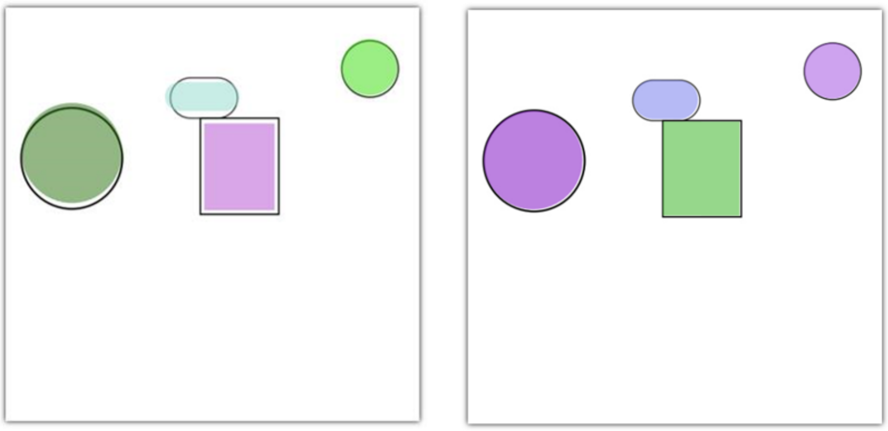
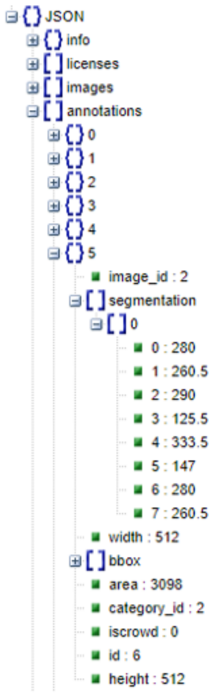


Figure 4.10: BestFit algorithm previously implemented result vs. *CAM2* BestFit algorithm result

the algorithms chosen and developed, since the perception obtained by looking only at coordinates does not allow a visual comprehension of how better the performance from a determined algorithm is.
As for the Retraining Service, the result achieved was the fill of the COCO dataset file used to train the model. A sample of this file is presented in the figure 4.11.



```
{} JSON
  {} info
  [] licenses
  [] images
  [] annotations
    {} 0
    {} 1
    {} 2
    {} 3
    {} 4
    {} 5
      image_id : 2
      [] segmentation
        [] 0
          0 : 280
          1 : 260.5
          2 : 290
          3 : 125.5
          4 : 333.5
          5 : 147
          6 : 280
          7 : 260.5
      width : 512
      [] bbox
      area : 3098
      category_id : 2
      iscrowd : 0
      id : 6
      height : 512
```

Figure 4.11: COCO dataset with the addition of a corrected technical drawing

4.2.2 Acceptance tests

The acceptance tests were developed in different scenarios, with different technical drawings. For the Recognition Service, the following types of technical drawings were tested:

1. Empty drawing;
2. Technical drawing without text information
3. Technical drawing with shapes that are not defined in the MaskRCNN model
4. Technical drawing with all the three shapes and text information
5. Technical drawing with arrows passing through shapes and connecting the information

The table 4.1 declares the test procedure to elaborate.

Test procedure	Accessing the *Technical Drawing Annotation Tool*, the user must insert one of the previously mentioned technical drawings, clicking on the recognition button, in order to send the drawings back to the server. On a different machine, the server must be running.

Table 4.1: Use Case: Technical Drawing Recognition

The table 4.2 compares the expected result with the actual result of these tests. For the lack of text recognition, as mentioned previously, it is due to the primarily stage of training of the OCR model and the existence of different types of characters in the PMI data. As for the deficient detection of shapes that contain arrows, the MaskRCNN model implemented is not yet prepared to receive this type of information, being unable to withdraw the correct shapes from these drawings.

4.3 Solution Assessment

The project was developed in an incremental process. This process helped evaluating the solution obtained and deducing the components that needed an improvement.
During the initial project weeks, the implementation of the gRPC framework was achieve through a Python client and a Python service, passing simple images and, in the service side, applying simple operations to the image, such as:

- Rotating an image;
- Converting the image to a black and white scale;
- Scaling the image;

Expected Result	1. No shapes, textboxes or text detected. 2. Shapes detected without visualizing textboxes. 3. Only the predefined shapes detected. 4. The shapes and textboxes detected. Text partially detected. 5. Shapes partially detected, as well as text.	**Obtained Result**	1. Success 2. Success 3. Success 4. Success 5. Success

Table 4.2: Expected result and obtained result for the Recognition Service use case

These operations allowed a gain of knowledge with some Python libraries, since there was no familiarity with the language before. After some simple operations, the client was implemented in C# and the connection was tested from different machines.
In order to apply the MaskRCNN model to the images received, there was a need of refactoring the given code and running it outside the Jupyter Notebook. The refactoring of the code involved creating classes and objects, allowing the model to be instantiated.
With the development of the *Technical Drawing Annotation Tool*, the client was implemented in this tool, allowing the communication between both internship projects. This advance led to the further implementation of the Recognition Service and the Retraining Service already addressed.
The incremental process allowed the manual testing of each functionality, operation and change made, producing the development path followed and the achievement of the solution presented.

5. Conclusions

In the actual development state, the system presents several advantages in its use. Since it is a proof of concept, in its primary stage, the technical drawings to be processed are simpler than the actual technical drawings found in this business area. Although, for the internship objectives,the goal was to build a service that would process simpler data, allowing, in the future, the increment of the performance of the used models, or even a change of these.

The solution developed also allows the incorporation of more services, with distinct functionalities and utility for the *Technical Drawing Annotation Tool*, since the microservices implemented do not depend on each other directly.

5.1 Results

The original objectives did not include the Retraining Service neither the use of the OCR model to recognize text. Both were achieved, although the performance of the OCR model is yet low. The table 5.1 summarizes the main tasks and states if they were achieved or not.

In the final solution, the Retraining Service is not fully working when connected to the *Technical Drawing Annotation Tool*, having connection problems related with the gRPC implementation.Also, the testing of the implemented services was not fully achieved, since it would be desirable end-to-end tests tests through both internship projects. Due to a lack of time and since the projects are separated, only the manual and acceptance tests were performed.

As for the concluded tasks, the model suggested to be applied in the text recognition still needs improvements for a better performance, as well as the MaskRCNN model, so that more complex technical drawings can be processed.

5.2 Limitations and Future Work

Even though the final solution is functional and allows an interesting use of the project aimed functionalities, it still has some limitations.

One of the limitations is the Python version and the libraries version needed to implement the MaskRCNN model developed by the Research Team. Since this model has

gRPC Web Service implementation	Achieved
Modification and implementation of the Mask-RCNN existent algo-rithm	Achieved
Development of best fit algorithms in Python (rectangles, circles androundslots)	Achieved
Implementation of the BestFit algorithm from CAM2	Achieved
Model corrections, manual tests and research about PMI	Achieved
Finding the syntax of PMI and defining 2D rules for 2D graphics	Partially achieved
Implementing PMI parser and writer	Achieved
Defining structures to include corrected data and loading old JSONfiles, update the details of the corrected shapes	Achieved
Implementing the model for text boxes detection	Achieved
Testing the implemented services to find bugs and correct them	Partially achieved

Table 5.1: State of the purposed tasks

been implemented in a previous Hackathon, it is clinging to older versions of Python and Tensorflow that already suffered bigger changes. This led to the need of creating an Anaconda Environment with the needed versions, preventing the use of other libraries that required newer versions.
Another limitation is the fact that the BestFit algorithm from CAM2 is implemented in C#, having to be called as an executable file. It comes up with a worst performance and versatility, since the algorithm needed changes to be used and does not allow the exchange of data between both processes.
The last limitation was due to the COVID-19 pandemic that led to a remote work environment and different way of working. Although the company response to the new reality, the lack of personal contact and discussion with the team made it more difficult to develop the work.
As a relevant future task, it is noteworthy the improvement of the MaskRCNN and the OCR models, expanding the training phase and database and changing previous model definitions, which will lead to a better performance and the achievement of more satisfactory results.

5.3 Final Appreciation

In retrospective, the author is satisfied with the project developed, although she knows that the complexity of the technical drawings to be processed can grow, leading to a decrease of the performance of the actual solution.
Planning and developing an application based on existing work can be a difficult task. Besides the need to refactor and change the previous work, there is also the need to understand what was previously achieved and how it is possible to go further.
Other difficulty was the use of an unknown programming language, as well as the introduction of Deep Learning algorithms and concepts that were being studied by the first time before. However, one of the main reasons why this internship was a preference was also its difficulty. There was a liking in being able to learn so many new concepts.
The incremental process applied led to a better understanding of the purpose of the project, even if, sometimes, it also resulted in major changes or enhancements.
Finally, the author would like to express the satisfaction of being part of this project, thank the support from the team and emphasize all the learning experience.

References

[DISCUS, 2006] DISCUS (2006). Discus desktop.

[Drumond, 2017] Drumond, C. (2017). Scrum.

[Eeles, 2004] Eeles, P. (2004). Furps+.

[Faro, 2018] Faro (2018). Software help sheet.

[Gandhi, 2018] Gandhi, R. (2018). R-cnn, fast r-cnn, faster r-cnn, yolo — object detection algorithms.

[Girshick et al., 2014] Girshick, R., Donahue, J., Darrel, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation.

[Grigorik, 2013] Grigorik, I. (2013). High performance browser networkingl.

[Guthrie, 2003] Guthrie (2003). Qa-cad software.

[He et al., 2018] He, K., Gkioxari, G., Dollar, P., and R., G. (2018). Mask-rcnn.

[Hui, 2018] Hui, J. (2018). Image segmentation with mask r-cnn.

[Indrasiri, 2018] Indrasiri, K. (2018). Build real-world microservices with grpc.

[InspectionXpert, 2004] InspectionXpert (2004). Inspectionxpert.

[Jaderberg et al., 2015] Jaderberg, M., Simonyan, K., Zisserman, A., and Kavukcuoglu, K. (2015). Spatial transformer networks.

[Koftikian, 2001] Koftikian, J. (2001). Simple object access protocol.

[Lorena and Carvalhor, 2002] Lorena, A. and Carvalhor, A. (2002). As maquinas de vetores suporte.

[Mikulecky, 2005] Mikulecky, M. (2005). Automatically balloning an assembly drawing of a computer aided design.

[P. F. Felzenszwalb and Ramanan, 2010] P. F. Felzenszwalb, R. B. Girshick, D. M. and Ramanan, D. (2010). Object detection with discriminatively trainedpart based models.

[Paradigm, 2017] Paradigm, V. (2017). Requirement analysis techniques.

[QA, 2009] QA, H. (2009). High qa inspection manager.

[Rational, 2001] Rational (2001). Rational unified process - best practices for software development teams.

[Redmon et al., 2015] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2015). You only look once:unified, real-time object detectione.

[RHe et al., 2018] RHe, K., Gkioxari, G., Dollár, P., and Girshick, R. (2018). Mask r-cnn.

[Riley, 2019] Riley, C. (2019). Know your api protocols: Soap vs. rest vs. json-rpc vs. grpc vs. graphql vs. thrift.

[Sandoval, 2018] Sandoval, K. (2018). When to use what: Rest, graphql, webhooks, grpc.

[Schwaber and Sutherland, 2013] Schwaber, K. and Sutherland, J. (2013). The scrum guide.

[Sharma et al., 2012] Sharma, S., Sarkar, D., and Gupta, D. (2012). Agile processes and methodologies: A conceptual study.

[Siemens, 2015] Siemens (2015). Product and manufacturing information.

[Vieira, 2015] Vieira, N. (2015). Splineapi, uma api rest paraservic os de processamento delinguagem natural.

[Wasson, 2019] Wasson, M. (2019). Microservices architecture style.

[Zhao et al., 2010] Zhao, Z.-Q., Xu, S.-t., and Wu, X. (2010). Object detection with deep learning: A review.

[Świeżewski, 2020] Świeżewski, J. (2020). Yolo algorithm and yolo object detection: An introduction.