Requirements

for

DAIKIRI

Version 3

Authors:

AI4BD	Jonas Haupt
Uni Paderborn	Prof. Dr. Axel-Cyrille Ngonga Ngomo
	Dr. Stefan Heindorf
	Caglar Demir
	Diego Moussallem
pmOne	Dr. Stefan Balke
USU	Dr. Carolin Walter

22.04.2020

Inhaltsverzeichnis

Introduction 1
- **Goal** 1

Use Cases 2
- **Use Case AI4BD** 2
- **Operating Environment** 4
- **Use Case USU** 6
- **Use Case pmOne (optional)** 7

External Dependencies for Interfaces 8
- **User Interfaces** 8
- **Software Interfaces** 8

System Requirements 10
- **Metrics** 10
- **Types of Machine Learning / Output** 10
- **Frameworks and APIs** 11
  - **Microservice-based architecture** 11
  - **Continuous Delivery and Deployment** 11
  - **Data Format** 11
  - **Human-in-the-Loop Accessibility** 12

Non-functional Requirements 13
- **Performance Requirements** 13
- **Produktsicherheitsanforderungen** 14
- **Datensicherheitsanforderungen** 14
- **Softwarequalitätsmerkmale** 14
- **Business Rules** 14

Further Requirements 14

1. Introduction

1.1 Goal

DAIKIRI aims to answer the following question: **How is it possible to create understandable ML-generated system diagnostics for domain experts with minimal effort?** The goals of DAIKIRI result from the achievement of the common goals:

1. **Efficient calculation of embeddings:** Usually, the scaling of existing methods for the calculation of embeddings (e.g. Word2Vec, HOLE) are superlinear in their runtime and space complexity. This means in particular that those with large input sizes (e.g. petabytes of data) can only be used to a limited extent. DAIKIRI develops a physical, linear method for the calculation of embeddings. Local and distributed implementations are used to process industrially relevant data volumes. The approach is compared with existing methods regarding its space and runtime complexity, as well as its performance in predicting classes for embeddings. For this purpose the method is combined with a novel clustering method.

2. **Scalable unsupervised and semi-supervised methods for learning ontologies:** For the learning process of ontologies three components have to be learned: classes, predicates and axioms. The above-mentioned embedding and clustering methods are combined with active learning, to learn classes semi-automatically. Class labeling is done using (1) unstructured data and (2) feedback from human experts. Novel methods will be developed for learning predicates in the embedding space. The aim is to develop a joint learning approach to combine human feedback and topological space properties. Axioms (i.e., OWL and RDFS axioms) will be learned with a combination of learning methods in the embedding space like Frequent Itemset Mining.
3. **Scalable, explainable machine learning methods for relevant sublanguages of SROIQ(D):** SROIQ(D) is, due to its power, a suitable language for learning complex models. However, this is associated with considerable runtime costs. Within DAIKIRI relevant sublanguages for the applications will be identified (e.g. EL++, ELQ, SHOIN).

Appropriate supervised learning procedures (e.g. refinement operators, version spaces) are developed and evaluated with benchmarks and application-relevant data. The verbalization of the learned models is carried out by dedicated procedures, which are developed in the project.

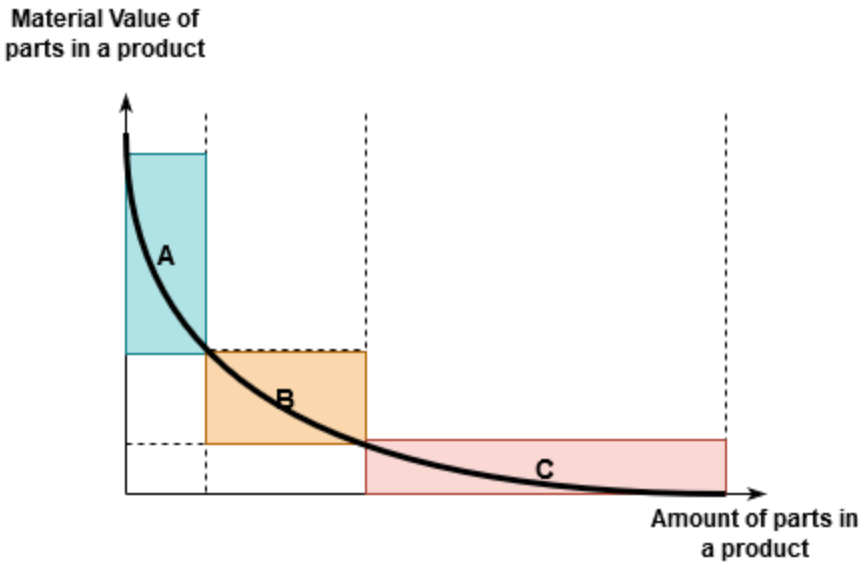
2. Use Cases

In this section, an overview of the use cases is given. Although, the initial cases are defined, they would be extended over the project runtime

2.1.1 Use Case AI4BD

A timely delivery of any parts to production lanes is crucial to have an efficient assembling of products - for serial and project-based production. It is required to have the **right parts** in a **sufficient amount** (not less but also not too much) at the **correct place** at a **given time**. Therefore, a proactive system is required which can predict the correct amount to deliver. However, there are a lot of obstacles to be solved, e.g., identification and elimination of anomalies, unclear production plans etc. Additionally, the results of the prediction need to be explainable and so judgeable if an AI-based system is doing well.

In today's fast moving world, manufacturing and production are the true drivers of growth. The better, the leaner, and the smarter you make your factory, the more likely you'll succeed in the market. As a logistic company, you have to manage A-, B- and C-Parts of thousands of customers, based on smart logistics systems. In this use case, we focus on B- and C parts. In product assembly, B parts make up to 25% of the costs overall parts and have an amount of 10-40% of all parts assembled. C-parts have only about 5-10% of the value but make up to 70% of all parts. Typical C-parts are screws or nuts. The graphic shows this relation.



As a logistics & inventory management company, you have the responsibility to provide the right parts:

- with the right quantity,
- with the right quality
- at the right time,
- at the right place.

Unfortunately, it is not possible to get a useful forecast from customers on B- and C-Part level. If one is available, either timing of usage or quantities are not correct. In other words, the bill of material (BOM) of the customers are incorrect or incomplete in relation to B- and C-parts.

The **main objective** is, to make use of the historic inventory data AND the future production plans of the customers in order to **predict** the required amount of B- and C-parts for the **next 2 replenishment cycles**. Since it is not feasible in one step and ILP is currently targeting classification tasks, the business objectives are:

Objective 1: Anomaly detection, classification and prediction

- Detection and classification of detectable anomalies

- Prediction of anomalies

Objective 2: Prediction of assortment consumption

- Prediction of the daily consumption of different assortment UUIDs
- Per customer
- Time range: 30 days
- Use the following evaluation strategy

Evaluation strategy

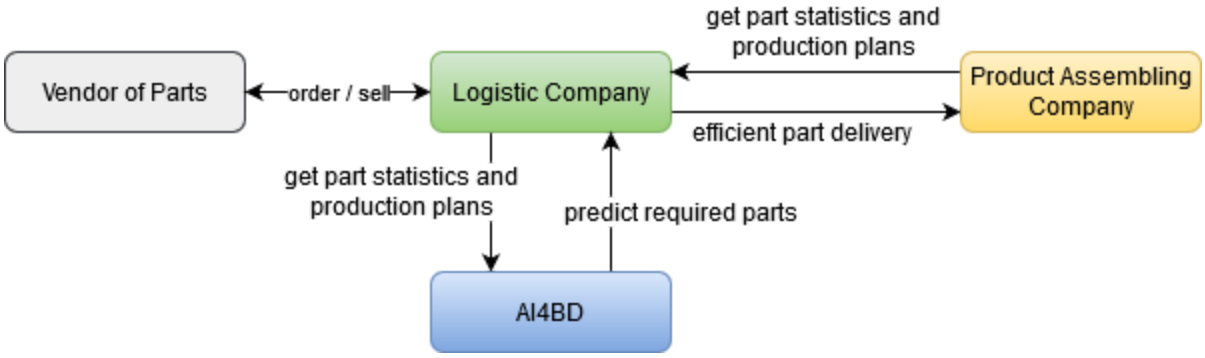
- **Service Level Agreement** (SLA per assortment UUID):
    - Upper bound: 10% of reorder point value
    - Lower bound: 5% of reorder point value
- Evaluation of data points per assortment UUID
    - Good: data points inside the SLA
    - Bad: data points outside the SLA
- Evaluation of predictions per assortment UUID:
    - Good: assortment UUIDs where all predictions are inside the SLA
    - Bad: assortment UUIDs where all predictions are outside the SLA

Target Groups

There are 2 target groups as users of the system. First, the company and their logistic experts, which are responsible to deliver the parts to the production lanes of its customers in-time. Occasionally, these experts need to validate the usually accumulated amounts to deliver. Second, the controllers of the production company are reviewing the quantity of the delivered parts. Both experts need to have explainable predictions.

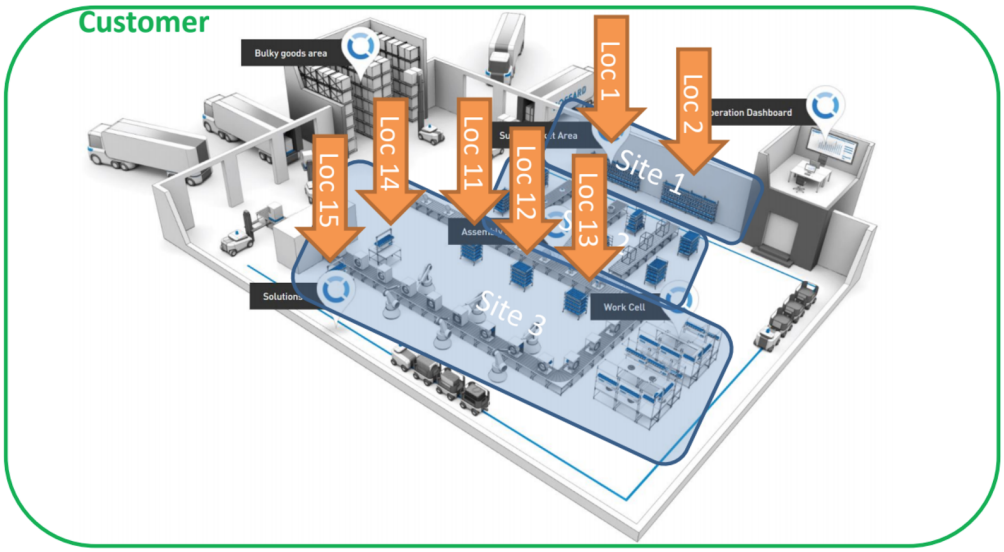
- **Operating Environment**

The following graphic gives an overview of the environment of the use case:



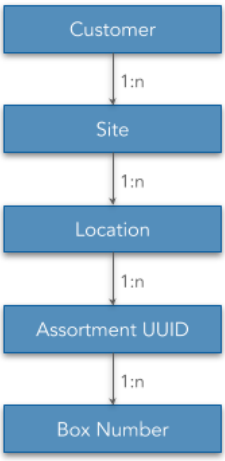
Overview of the resource hierarchy

- Definition of several resources:



- Hierarchy of resources:
  - Each customer has multiple production lines (**sites**)
  - each site consists of multiple boxes (**locations**)
  - each location contains multiple fastening elements of a specific type (**assortment UUID)**

- each assortment UUID can be distributed over multiple boxes (**box number**)



Determining the content of a certain box

- Known weight of a single assortment element (assortment UUID scale weight)
- Amount of elements inside a box calculated by total weight

2.1.2 Use Case USU

Use Case

Oscillating screening machines are used for screening, sorting and classification of coarse-grained materials. For this purpose, vibrations are placed on a screen, whereby different parts can be separated from one another. This process is used for example for recycling material.

With sensors the gear transmission function of these vibration machines is monitored in order to identify anomalies and to explain them. Events should be recognized and classified so that the algorithm recognizes symptoms. Example: A certain vibration that subsequently results in tooth breakage. Such breakage will not happen at short-term.

Business Understanding

Sensors on these vibration machines will only record overall oscillations. The basic oscillation is overlaid by the anomalies. The basic oscillation of the screening machines is 5-6 kHz. The anomalies are of high frequency. Therefore Piezo-Sensors are used, which operate at a frequency of 1 MHz.

Data

There are 10 to 12 sensors placed at different locations on the machine. The test system will work on different vibrations to figure out standard conditions against anomalies. There is a dataset for each of the sensors. It contains the ‘tickcnt’ and a decimal number for the data. ‘Tickcnt’ is an

integer with subsequent numbers for each scan. The scans are taken with the frequency denoted in the header of the dataset. The decimal number results from a Delta-Sigma-Modulation transmission and a 1-bit converter.

2.1.3 Use Case pmOne (optional)

pmOne's use case will consist of the development of an industrial IoT analytics testbed. To help simulate data close to a real production scenario, the work will be based on the "Fischertechnik Factory Simulation" Model, which consists of a small model of a factory with various sensors. Since the simulation is based on a production line, the IIoT end-to-end use case will be chosen adaptable to this setting. The architecture and data management platform will be set up in a cloud-based environment (e.g., Microsoft Azure) since it offers the flexibility to integrate state-of-the-art engineering and machine learning workflows.

At first, the connection between the Fischertechnik Factory Model and the Azure cloud environment will be set up. Furthermore, additional sensors will be connected to the Factory Model and to the Azure cloud. After having a proper data connection of all needed sensors, a data exploration and data preparation of the sensor data will be carried out.

In an experimental phase, specific errors will be introduced to the model to enforce anomalies in the simulated production process. This data will then be used to train initial machine learning models which in turn can be used for the explainable AI work packages.

3. External Dependencies for Interfaces

3.1 User Interfaces

The goal of DAIKIRI is to make the results of ML-based diagnosis of industrial data explainable to users, who are not familiar with ML. This is achieved by learning OWL DL expressions which explain the predictions and by verbalizing the expressions.

3.2 Software Interfaces

After cleansing the data, the data is transformed to an embedding space. Based on the clusters, an ontology is learned in a semi-supervised manner.

Data Preparation

Industrial data is given in tabular form (e.g., as CSV files or SQL dumps) and must be cleaned by the data providers (AI4BD, USU, PmOne). The data preparation includes:

- Identification and removal of redundant information, e.g. entities that are represented by multiple redundant attributes like an ID, a number, a name, a timestamp
- Merge of data from different sources/files
    - Define rules to identify data samples from different sources / files that belong together
    - Combine data samples based on the formulated rules
    - Combine data samples based on common timestamps
- Annotation of classes (e.g. detectable anomalies)
    - Definition of rules describing the classes - each class need to be represented by one or multiple rules
    - Calculation of value-added data required for the desired task
    - Applying the formulated rules on the data to label classes
    - Annotation based on video-data
- A potential tool to support the data cleansing might be OpenRefine (https://openrefine.org/).

Transformation for Embeddings

Based on the cleaned tabular data, embeddings are computed (without prior transformation to RDF).

Semantification

Based on the tabular data and the embeddings an OWL ontology is generated semi-automatically: RDF resources are clustered automatically, labels for the clusters are specified manually where necessary, and the ontology is enriched with axioms.

For the manual annotation a user interface is required that allows to describe the ontology with human-readable properties that are later used for verbalization (e.g., `rdfs:label`, `rdfs:comment`, `rdfs:seeAlso`,`rdfs:subClassOf`, `skos:definition`).

Based on the learned ontology, the tabular data is transformed to RDF.For example, rows are converted to ressources, columns to properties, atomic values to literals, and foreign keys connect resources.

Explainable Machine Learning

Given RDF and OWL files from the semantification as well as positive and negative training samples, OWL2-DL expressions are learned to discriminate between positive and negative examples.

Since OWL2-DL expressions might not be easily understandable by domain experts, they are verbalized in the form of natural language. For this, a multilingual, rule-based approach will be developed. The languages English and German will be supported.

4. System Requirements

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

4.1 Metrics

Based on the use case problem, algorithms should be able to give different weights to measures such as precision, recall or accuracy.

- Precision, Recall, Accuracy and F1-Score:
    - We apply Precision P = TP / (TP + FP), Recall R = TP / (TP + FN), Accuracy A = TP + TN / (TP + FP + FN+TN) and F1-Score F1 = 2 * P / (P * R). TP, FP, FN, and TN are determined as defined in the DL-learner framework [1]. Our main measure will be the F-measure. Target: We will aim to outperform the state of the art on selected benchmark datasets. As current performances vary considerably depending on the benchmark, we will aim to achieve an F-measure which reduces the gap to the perfect classification (ergo, 1-F-measure) by 10% on average within time-restricted experimental settings.
- We will evaluate the embeddings in terms of Hits@n and the area under the receiver operating characteristics (AUC).
- We will evaluate the verbalization with semantic similarity measures to measure the intentional similarity between target concepts and learned concepts (e.g., BLEU, METEOR and chrF3).
- The input of the ML is data in RDF format and an OWL ontology.
- The data needs to be accessible via a file API or via a SPARQL endpoint.
- The algorithm needs to be configurable via (nested) config maps. Example parameters are for example the input source and the training data.

4.2 Types of Machine Learning / Output

- The ML component needs to be able to solve classification questions.
- The ML component may also solve regression questions.
- The output (the answer) to the ML component is 1 to n OWL axioms. They can be returned directly to the requester or stored in a predefined target.
- The output comes with confidence scores as defined in 4.1 metrics to understand the quality of the results.

4.3 Frameworks and APIs

4.3.1 Microservice-based architecture

- Every "business" functionality needs to be decomposed to re-usable services in order to be developed and maintained in a decoupled way. Each service has its own tests.
- The functionality needs to be available as a web service using a REST API.
- The REST API needs to have a well-defined OpenAPI file including the data schema.
- Data management is decoupled from a service, they are stateless
- To have proper ETL pipelines on-top of the microservices, existing frameworks need to be used.

4.3.2 Continuous Delivery and Deployment

- Each service needs to be package as a Docker images
- There need to be proper build pipelines to create the Docker images from software artifacts.
- The Docker images need to be available for all partners by a proper registry.
- The testing and benchmarking should be automated.

4.3.3 Data Format

- RDF (Resource Description Framework) is the data format to work on. The serialization format is open and up to the implementation, however,
- JSON-LD is recommended to be useable in all web application

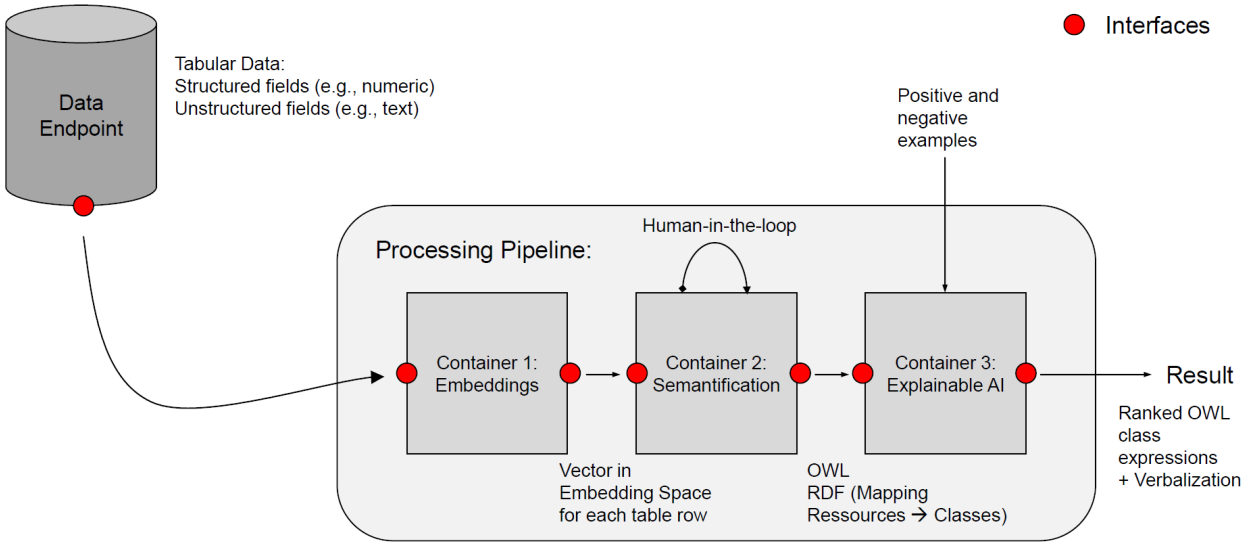
- TTL is compressed and nice to read for humans
    - N-Triples is good to read in chunks.
- SPARQL as query language on RDF
- The ontologies might be defined in RDFS and / or OWL.

4.3.4 Human-in-the-Loop Accessibility

- Reachability of a domain expert has to be secured with a guaranteed maximum reaction time. Concrete requirements can be on a case-by-case basis synchronous or asynchronous.
- Communication and involvement with the domain expert have to be consistent with the GDPR.

4.4 Platform

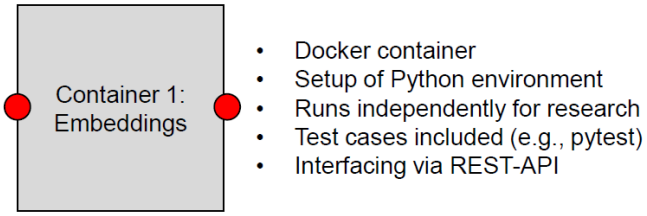
4.4.1 Overview



- The platform consists of a data endpoint and a processing pipeline.
- The data endpoint can either be SPARQL-based or offer RDF files.
- The processing pipeline mimics the different processing steps. Namely: Embeddings calculations, Semantification, and Explainable AI.

- The data between the processing steps is realized via standardized and documented interfaces.

4.4.2 Processing Step



- Each processing step in the pipeline is a containerized application (e.g., Docker).
- The container management system establishes the development and/or production environment, supporting reproducibility of experiments.
- Each container, with the necessary input data and configuration, is meant to run independently. Especially in phases of research, this requirement needs to be fulfilled to enable researcher easy access to the pipeline step.
- Each container contains procedures for test cases. However, data for the test cases might come from an external data endpoint.
- Interfaces should allow to easily access the container's functionality by using REST-APIs.

4.4.2 Scalability

- For production environments, flexible scalability (e.g., through Kubernetes clusters) is a requirement.
- However, simplicity and pragmatic choices to enable research is the priority in this project.
- Transforming the processing pipeline to a scalable pipeline lies in the responsibility of the respective use case owner.

5. Non-functional Requirements

5.1 Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

5.2 Produktsicherheitsanforderungen

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.>

5.3 Datensicherheitsanforderungen

Data providers (e.g., USU, AI4BD, PM1) have to anonymize the data before providing them to other partners (e.g., UPB). This might be achieved by replacing person names with hash values.

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

5.4 Softwarequalitätsmerkmale

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

5.5 Business Rules

<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>

6. Further Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Glossary