WIRTSCHAFTS
UNIVERSITÄT
WIEN VIENNA
UNIVERSITY OF
ECONOMICS
AND BUSINESS

Towards a Workflow-based AI System Documentation

Master's Thesis

Bernhard Kollmann
11817868

Department of Information Systems and Operations Management
Institute for Digital Ecosystems

Advisor:	Fajar J. Ekaputra
Second advisor:	Laura Waltersdorfer

30th of September 2024

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
BEAM	Boxology's Extended Annotation for Machine Learning Systems
BPMN	Business Process Model and Notation
CLI	Command Line Interface
DA	Data Requirements
EC	Ecosystem Requirements
ENT	Ear-Nose-Throat (Physician)
ET	Ethical Requirements
ETL	Extract-Transform-Load (Process)
LLM	Large Language Model
ML	Machine Learning
RDF	Resource Description Framework
*RQ*1	Research Question 1
*RQ*2	Research Question 2
*RQ*3	Research Question 3
ST	(Documentation) Structure
TMmodelling	Thinging Machine modelling
UI	User Interface
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WF	Workflow Requirements
XAI	Explainable Artificial Intelligence

Abstract

In the last decade, the emergence of AI and systems incorporating AI as a core technology has risen substantially. Recent advances make it possible to make general queries to large language models or to create music based on input. However, it is currently a challenge to document these systems correctly and comprehensibly due to their black-box nature. In the future, sound documentation will be mandatory due to the upcoming legislation such as the EU AI Act. To date, there is no standardised way of documenting AI systems has been generally accepted in the community. Therefore, this thesis aims to advance research on this challenge. Specifically, this thesis is concerned with exploring the extent to which automation based on workflow documentation of AI systems is possible. To this end, several steps are necessary. First, we need to identify the stakeholders and their requirements. The next step is to find out how AI system workflow can be documented in a way that sufficiently meets the identified requirements. The final part is to answer the question of how to support automation in representing the workflow of AI systems. Our study has identified six distinct stakeholder groups and their requirements. Further, we found that although several suitable candidate approaches for documenting AI systems workflow were identified, none of them was able to meet all the stakeholder requirements. Finally, we have developed a proof of concept to support automation regarding AI system workflow documentation consisting of two parts: (i) a proposed extension of an existing notation and (ii) a toolkit to enable automated transformation of the workflow notation to its machine-readable representation. Our evaluation shows promising results, which serve as a solid foundation for future developments.

1. Introduction

Proper documentation of software can be challenging, especially for complex software projects. The difficulty will be amplified further when an AI component is included, as the operation of the AI and the surrounding ecosystem need to be considered. Even if the documentation is created properly, other readers may not take the time to read the documentation adequately [CC22]. However, the situation is different in scenarios where exact, unambiguous documentation that is fully comprehensible for each involved party is mandatory. Legislation, such as the *EU AI Act*, is pushing companies to accurately present how AI products work, how they comply with existing laws and regulations, and what safety measures are taken if they are considered risky [EuA24a]. Furthermore, the legal requirements situation is only one of many perspectives to consider. The facts that are apparent to the AI system developers may not be so obvious to the legal auditor who has to decide whether an AI system is safe enough to operate beyond a certain level of development. Moreover, there may be a significant difference in the expected response to an enquiry about such a system, even if the same question is asked. For example, a user inquiring about the operation of the AI system may ask how the data provided may be processed. Conversely, a researcher in the field may ask how the AI model works and how some specific challenges have been overcome [HMD$^+$23].

Previous work has attempted to address this challenge by adapting general approaches to software documentation to handle the additional challenges of documenting AI systems. These challenges include the difficulty of documenting AI itself, as well as its environment. Moreover, they proposed different solutions that apply to different documentation readers. For example, to meet the needs of parties requiring more thorough documentation of AI and its ecosystem, various AI system models have been proposed and used [HG22, JPZ22, ELS$^+$23a, vdML02, SL04]. Thinging Machine Modelling [AF21] proposed to combine different UML models to provide a more concise overview in one diagram. Other approaches are focus on the stakeholders requiring more flexibility and granularity when modelling AI systems used ontologies or data cards [NMEC21, ELS$^+$23a, SRMP$^+$22, DG23, PZK22]. Lastly, more specialised approaches have been detected as well. Specifically for code documentation, an option was found that prompts the code to an AI model and automatically generates comments [WWD$^+$22].

This thesis aims to contribute to addressing the challenges above and enrich the discourse on the topic of AI system documentation. Specifically, we aim to answer the main research question:

To what extent can the automation of AI systems documentation be supported through system workflow representation?

Due to its complexity, multiple steps are required to answer this question. We elaborated on these steps and defined separated sub-research questions. The first step encompasses the collection of evidence about all the interested parties and their requirements for the documentation. Thus, the first sub-question (RQ1) is formulated as: *"Who are the stake-*

holders, and what are their requirements for AI systems documentation?". To answer this question, we have conducted an extensive literature review on the topic.

For the second step, we aim to explore how to meet the identified requirements concerning AI system documentation. As such, the second sub-question (RQ2) would be: *"How to represent AI system workflow to support AI systems documentation?"*. To answer this question, we explored and evaluated existing notations, focusing on their ability to meet the requirements found in the first step. Primarily, we will consider works that have used particular notations to describe AI system workflow. Furthermore, to gain a better insight into the field, we also extend our scope to other recent works dealing with AI system documentation.

Finally, the last step is to uncover to what extent automation can help the documentation process of existing AI systems. The sub-question (RQ3) has been formulated as: *"How to design and develop tool supports for representing system workflow in the context of AI system documentation?"*. To answer this sub-research question, we developed a set of artefacts based on an extension to the existing work of the Boxology notation presented by van Bekkum et al. [VHTT19].

The developed artefacts, called Boxology's Extended Annotation for Machine Learning Systems (BEAM), are based on the requirements identified of two specific stakeholder groups: *producers* and *academics*. BEAM consists of two parts: (i) the extended boxology notation, where we compared the requirements of these specific stakeholder groups with the possibilities offered by the original boxology notation to address these stakeholders' requirements sufficiently, and (ii) the BEAM toolkit, aiming to automatically extract a diagram created in the extended notation, restructure the data and possibly represent it in a different format, such as an ontology, validate the syntax and semantics where applicable, and allows querying of the data. Furthermore, a first step has also been taken to provide public access to this service through a small Flask application [Gri18].

Our result shows that support for AI system documentation automation can be achieved, and various tasks (e.g., storage, querying, validation of AI system workflow) can be conducted with the information stored within the diagrams. However, multiple pre-conditions have to be fulfilled to understand and use the data in the context of scripts that work with that data. First, the essential aspects of documentation and the documentation of AI systems need to be defined. Furthermore, there is a plethora of requirements from different stakeholders, such as legal entities, producers, users, academics, bystanders, or ethic activists (RQ1). The requirements have been broadly categorised into requirements for the workflow itself, requirements for the ecosystem, requirements for the data used, requirements for the structure of the documentation and ethical requirements.

Further results from our literature study on RQ2 have shown that several methods exist for representing AI system workflow. However, none can sufficiently satisfy all the identified requirements from RQ1. Consequently, to allow a greater degree of automation that satisfies as many requirements as possible, a notation that is better suited to satisfy these requirements is needed. Thus, to meet this requirement and answer the last sub-question

RQ3, we proposed an extension of the boxology notation [VHTT19].

We evaluated the notation extension through a survey, showing the usability and usefulness of some extensions, including a legend, predefined elements, or the ability to leave comments on elements. Other features, such metadata attachment to a workflow item, were considered beneficial but did not receive as much attention as the aforementioned items. Conversely, however, some additions, such as lists of enumerated features, were considered to be convoluted or confusing. The second contribution of this thesis is the BEAM toolkit. It supports AI system workflow documentation by allowing data manipulation, automatic restructuring of the data, querying the data, validating the content of the diagram, and offering these tools as a service in a web-application. Thus, to return to the main research question, we demonstrated that automation could support the documentation of an AI system in different ways, given that the AI system workflow diagram is represented in a machine-readable format and the diagram and the symbols used can be interpreted correctly, both by users and software.

The rest of the thesis is divided into the following chapters. First, a brief overview of the literature review process is provided in the related work 2. Following the related work, the research structure and approaches to answering the research question are presented as the research methodology 3. Next, attention is turned to the findings. The following sections illustrate how the research questions are answered. First, the identified user requirements are synthesised in chapter 4. Second, a comparison of different visualisation tools is presented and the rationale behind the choice of the tool used for the upcoming tasks is given. This is done in chapter 5. Thirdly, the developed tool, its usage, features, advantages, disadvantages and technical limitations are described in detail in chapter 6. As for the next set of sections, the discussion about the implications of this work is provided in chapter 7. Subsequently, the limitations are shown in chapter 7.5. Finally, the thesis concludes with a conclusion, and a future outlook can be seen in chapter 8.

2. Related Work

Before delving further into the field revolving research, insights from already gathered knowledge in the field are to be considered. To answer the research questions adequately, a couple of steps are required. As the upcoming examples will show, numerous approaches to document AI systems have been inspired by ”general” software documentation [EuA24a], [HFLBG24]. Thus, the approaches to document and their differences are elaborated first. The next step would be to narrow down the focus on AI systems and particularly elaborate on the differences to regular software in terms of documentation. Finally, the found options options currently used to document AI systems are to be elaborated.

2.1. Software Documentation

Writing down descriptions on auxiliary material to transmit information has always been important to store knowledge and transfer it among individuals. However, this is not always an easy task, especially when each individual has to understand the information to be learnt exactly, without leaving any room for interpretation. This is no different for software, and is perhaps even more important in this context. Software presents the challenge of being quite difficult to communicate correctly, especially in the case of larger systems. To meet these challenges, means of abstraction and simplification, namely models, have been developed. Currently, the use of visual aids in the form of diagrams and additional written descriptions appears to be the primary method of documenting software at a larger scale [KPL+14] [HFLBG24].

Using visuals has been a solution to the problem, which has been proven effective in the past [HFLBG24], [CDVR22], [CC22]. However, as the vast selection of different existing notations currently used may indicate, there is considerable scope for debates about the ”correct” approach. The information that needs to be depicted depends on several factors, such as the domain of application, the target audience, the exact purpose of the software, or even just to adhere to legal requirements and restrictions. One example of such restrictions would be the EU AI Act [EuA24a], [CBB+02].

Even when considering all these factors, multiple commonly used notations still would be able to accomplish the task adequately. Thus, the found literature diverges and numerous answers are more argued via the definition of functional requirements for specific target stakeholders and general non-functional requirements [Mil22], [WBD+21]. However, little evidence has been found of a general overview of all the requirements arising from different stakeholders, thus providing insight into multiple perspectives.

In terms of content for the documentation, many additional points are brought up into the discussion, as matters like granularity of the information, length of the content, or sensitive information have to be included in the system description as well [KT21], [BAW+20]. Documentation has become an artefact not just for the development team. One effect of this movement is the emergence of new requirements that need to be addressed. Going back to the multiple perspectives mentioned earlier, even the same requirement may have additional differences depending on the target audience [KT21].

2.2. AI Model- & AI Systems Documentation

Narrowing down the focus on current insights on documentation AI systems specifically, various notations and options to model AI systems have already been recommended in existing works. However, firstly, the challenge of documenting AI itself is to be discussed.

Unlike the documentation of most other software, AI differs in a few ways. Although this is a crude description, AI models typically work by taking data, passing it through functions that take into account weights and biases, and calculating an output. This output is forwarded if a certain threshold is exceeded, or ignored if not. This process is usually repeated multiple times. Finally, when the final layer is reached, the result is compared with the actual result and the said weights and biases are adjusted depending on how far the predicted result was from the actual result. It should also be noted that regardless of the original input type, the whole system operates on numbers and that the process described above is usually repeated, possibly hundreds or thousands of times. Although technically possible, it is difficult to describe how the predicted result was calculated, as this would mean following every number of every cycle to give an accurate answer. So there is a huge field of research into finding ways to better explain these results, called *explainable AI* or *XAI* for short [DR20], [GJS22].

Scaling this up to the scales inhabited by some systems built around the use of AI introduces a new set of challenges. AI systems generally describe an ecosystem built around the use of one or more AI models. Within this ecosystem, other elements are introduced that are critical to its operation. First, in many cases, the flow of data is covered. Data usually refers to the direct data used as input for the AI model(s) and the results produced. Sometimes the data preparation process is also included in the description. Where this is the case, data may also be used to highlight intermediate results at this stage. In addition, some AI models make use of pre-existing models, either during the training process before the input is fed into an AI model, or afterwards. The use of knowledge resources beforehand may arise in the scenario where the data needs to be mapped first. Conversely, the use of external knowledge resources after the fact is used, for example, to query existing databases [MZM21], [VHTT19], [ELS$^+$23a].

2.3. Current Modelling Approaches for AI systems Documentation

At this point, the question on how to best convey the workflow of on AI system concisely and unambiguously to other parties interested may have arisen. Apart from textual descriptions, one of the most common approaches was to use already tested and proven diagrams. One of the most frequently occurring group of diagrams would be different UML diagrams. UML provides a plethora of different diagrams specialized in displaying different perspectives. As such, some of the notations are also considered helpful for displaying AI systems. However, some of the research has pointed out that using UML for AI systems has had problems in modelling the system exactly as it works, due to issues such as insufficient element coverage [HG22], [JPZ22],[ELS+23a], [vdML02], [SL04]. In practice, UML models have been used to design hybrid AI systems by combining different "traditional" methods with AI. Other practical examples entail the modeling of the implementation of AI solutions in the healthcare sector [JPZ22], [BHK+24].

Research has also attempted to overcome the obstacle of too much rigidity by combining multiple UML models. This allows the same system to be viewed from multiple perspectives, highlighting the strengths of each and overcoming potential limitations of them. However, this tactic usually involves a significant increase in workload and new challenges such as inconsistencies between perspectives and misunderstandings due to the additional attention and understanding required to process all the models [AF21]. To overcome these new problems, *Thinging Machine Modelling* (or TM modelling for short) has been introduced. TM modelling is based on the notation of UML models and builds a "meta-model" consisting of said multiple models [KPL+14], [AF21].

Another option being explored is ontologies. Ontologies are a formal naming description of knowledge relationships. They are typically represented by a graph. Usually, ontologies link related concepts with verbs describing the relationship between them. The linking of content between them is flexible, which means that there are usually no strict semantic rules to follow in choosing the description for that link. As such, ontologies are a flexible structure for linking information [NMEC21], [ELS+23a], [SRMP+22], [DG23]. An example of an ontology would be *Rains*. Rains has been developed to represent concepts in AI systems, in particular [NMEC21]. Using ontologies to represent the AI system has been used as a method to make the components understandable to other researchers, as well as using languages such as SPARQL to query these components [ELS+23a]. Other studies have used ontologies to represent the accountability for AI systems as well as all parties involved [NMEC21].

A quite different approach entails boxologies. Boxologies may be viewed as a type of flow-diagram using simple shapes to represent different elements necessary for the operation of an AI system. Originally, the notation was designed to represent AI models incorporating symbolic or neuro-symbolic approaches. The design is set to cover various levels of abstraction. As such, it is entirely possible to combine different sub-systems into a composite model. Furthermore, inspiration has been drawn from object oriented programming to concisely represent scenarios like class- and instance-relations. To cover this, the authors specified that the boxology is designed to be used for both, creating "blueprints" of a system, as well as the actual instance with the very same elements. As described by the authors of the literature, the boxology was designed to better represent hybrid AI systems and foster communication among peers. Within the work, the notation has been tested in two use-cases. Firstly, it was tested in the form of modeling an AI system witch matches skills of applicants against the job description. Secondly, the boxology was used to model a system which predicts the best action for a robot in various scenarios [VHTT19].

At this point, a few approaches for modeling have been presented. However, other methods of documentation have been found as well. Firstly, a concept called data cards has been used for AI. Data cards provide different sections covering aspects, such as the general purpose, the workflow, and different benchmarks for instance. The content and segregation of each card can be adapted as needed. Thus, using data cards usually provides flexibility [PZK22]. Another entirely different approach called *Themisto*, which is able to work directly with the code and extract the required information autonomously [WWD$^+$22]. Via an own AI model, the system allows users to prompt their system to create an automatic documentation based on pre-defined texts. The AI model predicts the combination of what it believes to be the most appropriate responses to the prompt. However, this technique is only focused on documenting the process of the code. Nevertheless, it may prove to be a notable asset in boosting production in the future. Currently, it has been tested on widely used notebooks posted on Kaggle [WWD$^+$22].

3. Research Methodology

As for all scientific works best practices are to be followed and applied. Given the still rather exploratory scenario to which this work is applied, this work in particular is designed to produce artefacts in the form of additional knowledge as well as applicable proofs of concept. Consequently, the thesis follows the design science approach [VBHM20]. To apply the teachings in practice, Figure 3.1 elaborates on said application in more detail.

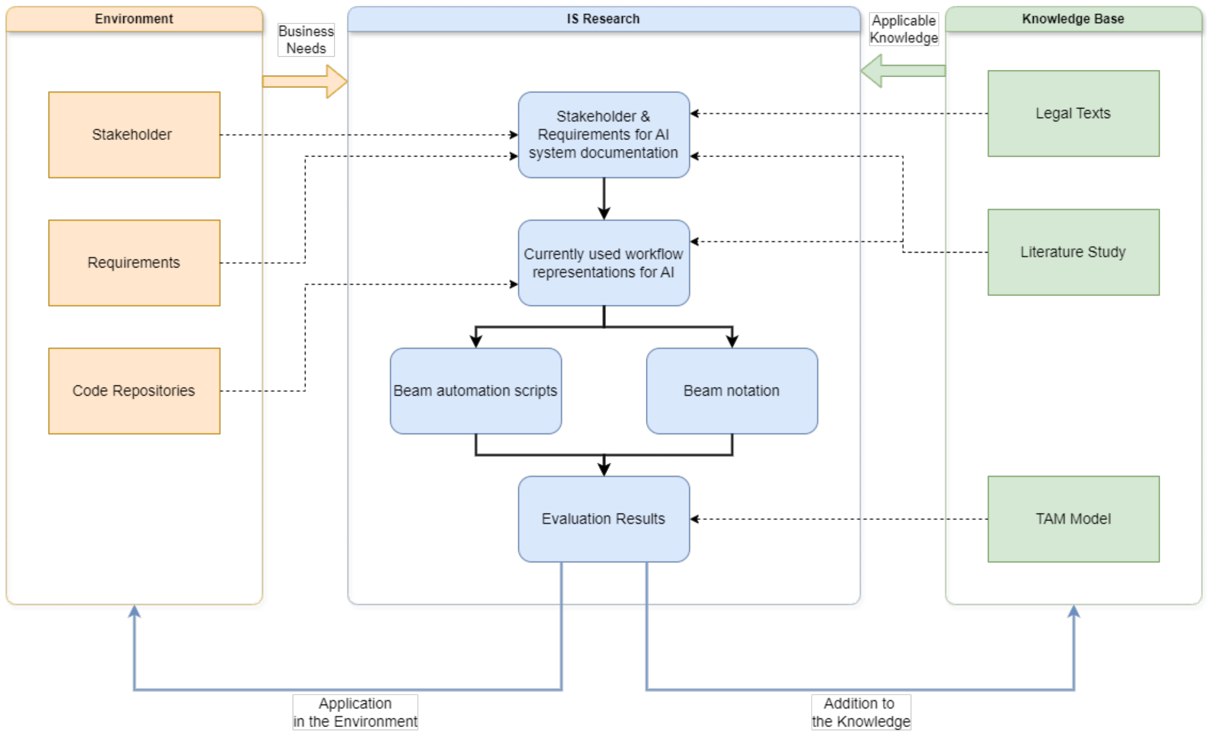


Figure 3.1.: Design science approach [VBHM20], [ELS+23a]

To answer the research question *"To what extent can the automation of AI systems documentation be supported through system workflow representation?"*, three underlying subquestions with different scientific practices will be utilized. Each of the question can be found in Table 3.1. For the first question (*RQ1*), the objective is to find different stakeholders and their requirements concerning the documentation of AI systems. Thus, finding answers in already existing work will be used as approach to answer this particular subquestion.

Advancing to the second research question (*RQ2*), to find the most promising notations to cover the found requirements, a similar approach to the previous one can be used. In this case, existing works linked in literature or found on well-known domains featuring the storage, presentation and sharing of such content will be scouted.

For the final question (*RQ3*), a practical approach was pursuit. As such, the requirements of two stakeholders will be used as starting point. A suiting notation will be chosen to use as the basis for attempts on automation, as well as meeting the requirements for the mentioned stakeholders. These latter mentioned requirements may entail the extension of existing notations. To get an initial response on how useful easy to use the extension is, a survey is conducted. Finally, to cover the automation, a proof of concept will be developed. This proof of concept will be elaborated as a concept of its functional components, as well as in the form of a description of the actual implementation. The implementation itself is based on the requirements found for the two stakeholder groups are elaborated in detail in section 6.3.4. Additionally, some participants of the survey also kindly provided their modelled AI systems. The extracted data of these models was used to test the proof of concept in its functionality.

Research Question	**Method to Answer**
[*RQ1*] *Who are the stakeholders, and what are their requirements for AI systems documentation?*	Literature research of academic literature legal texts, and other references such as code repositories
[*RQ2*] *How to represent AI system workflow to support AI systems documentation?*	Literature research Comparison of existing approaches based on the findings Checking code repositories
[*RQ3*] *How to design and develop tool supports for representing system workflow in the context of AI system documentation?*	Developing an extension of a notation Evaluation of said notation via a survey Developing a proof of concept

Table 3.1.: Approach to answer the research questions

With the general introduction to the research questions covered, the remainder of this chapter focuses on further elaborating the methods used to answer said research questions.

3.1. Literature Research

As with most scientific works, a solid foundation of literature is required to answer the research questions. Especially for the current thesis, a substantial amount of comparison between existing literature is required. As such, a literature review has been conducted. For each of the topics addressed in the thesis, certain key-phrases have been developed and used to find apt sources. For example, the key phrase *"Stakeholder of AI documentation"* has been used to detect resources concerning the first sub-question.

The results of the literature research can be found in Appendix A. In terms of finding appropriate literature and references, the following databases were used to detect apt literature [RS04]:

- Legal acts and related documents
- Google Scholar [Goo24]

- Emerald [Eme19]
- Scopus [Els23b]
- WU Library [Wu.23]

In addition to the scientific databases, other sources of information have been utilized as well. In particular, for finding different notations used to depict AI system, as well as for the development of the prototype, different code repository platforms, such as *GitHub* have been checked as well [git20].

3.2. Comparison of Notations

This measure relies on the results of the literature research described in the previous section. The found notations used to document AI systems will be evaluated against the detected requirements. The results shall subsequently be used to pick one option and tailor it further towards the requirements found.

3.3. Developing an Extension of a Notation

To develop an extension of an existing notation is based on the idea that current approaches do not cover all existing requirements sufficiently. As such, based on findings from the literature and the comparison of different notations currently used, a specific portion will be used as a baseline to develop one notation further. The aim would be to provide an improved notation that is better suited to the needs of modelling AI systems.

3.4. Evaluation of the Notation

In order to obtain a first impression on how the developed extension would fair in practice, a survey has been conducted. The participants were asked to try the extension of the notation and subsequently fill out a questionnaire. The questionnaire itself will be derived from best practice examples used to evaluate software.

3.5. Proof of Concept

Finally, to conclude the work and answer the question of possible automation, a toolkit working with the developed notation extension has been developed. This toolkit is supposed to act as a proof of concept by demonstrating how automation would work in practice. This step is set up into two different sub-steps. Firstly, a conceptual description of the functionality will be provided. Based on said description, future works can develop different toolkits based on the ideas. Secondly, the actual implementation is described in technical detail, with the objective of allowing future work to develop the proof of concept further. The implementation of the toolkit has been done in Python and additional third party libraries [VRDJ95]. The details can be found at the accompanying repository of this work [1]. As for the evaluation, the requirements of the aforementioned stakeholder groups,

[1] `https://git.wu.ac.at/semsys/master`

namely the *producers*, and the *academic sector*, have been used as a baseline for the evaluation, as these two groups are considered most likely to potentially continue working on the approach of this work. The evaluation takes the form of working out how the requirements have been met in the discussion 7.

4. Stakeholders Requirements for AI Systems Documentation

With the illustration of the current status covered, the attention can be redirected to the results. Following the presented order of the research questions, first the target stakeholder of AI system documentation, and their requirements are to be illustrated for *RQ1*. As the result is to highlight the various stakeholders interested in AI system documentation, as well as their respective requirements subsequently, a comprehensive literature review has been conducted. First, the various stakeholder uncovered will be presented. Following, the requirements are elaborated in more detail. Finally, this section will conclude by mapping the stakeholders to the requirements found. The results close by presenting an overview of all found stakeholders and their requirements in Table 4.2, as well as Table 4.3 respectively.

4.1. Stakeholder of AI-System Documentation

Starting with the stakeholders, various terms and roles, referring to the same stakeholder, have been found. Thus, should this case occur, the different terms will be summarised under a general term, but mentioned in the description by the found original term.

Furthermore, to structure this subsection, each stakeholder has been categorized into different stakeholder groups, such as *Legal Entities*, *Producers*, *Users*, the *Academic Sector*, or *Ethic Activists*. While the initial approach for formalising stakeholder was deviated from the EU AI Act, some definitions, such as *"distributor"* did not fit the narrative of other found literature too well, as it is an description for a provider offering any form of AI on the European market [EuA24b]. However, this definition would exclude private individuals publishing AI system as well as researchers. Thus, the following definitions of stakeholder groups were derived from [BAW$^+$20], [HMD$^+$23], [PHB$^+$18], [MHDSG23], [WBD$^+$21], [BHKST22], [GJS22], [HCHVD21].

4.1.1. Legal Entities

Starting with legal entities, they refer to all stakeholder concerned with regulating AI systems and their usage in any way. T This can be achieved through legislative means, such as the introduction of legislation, the assessment of such systems through audits, or the enforcement of existing laws and regulations.

- **Legislative/Policy Makers:** Legislative representatives in the context of this work refer to every individual or party responsible for proposing new laws, altering or extending existing laws concerning the usage of AI systems. One prominent example of such a body of regulations would be the *European AI Act.* [EuA24a],[KT21], [KT22], [HMD$^+$23], [MZM21], [RRF$^+$22], [LWM24]
- **Executive:** The executive acts as a collective term in itself, as the assessment and enforcement are included. Concerning the assessment in the scope of AI systems,

specifically legal audits are referred as key stakeholder. This party is responsible for assessing the exact risk of AI systems, if an AI system is considered to be at least a limited risk system under the definition of the EU AI Act [EuA24a]. Furthermore, assessment includes to evaluate situations in which potential damage (for example breach of data privacy involving an AI system) has been done. Finally, the law enforcement would be responsible for imposing the according legal consequences [EuA24b], [KT21], [RRF$^+$22], [PHB$^+$18], [HMD$^+$23].

4.1.2. Producers of AI Systems

This group includes all parties involved in at least one step of the development/production of an AI system. This definition includes companies that intend to use an AI system for commercial purposes, as well as individuals who publish an AI system on any platform accessible to at least one other party. However, the *Academic Sector* is separated into its own subsection 4.1.4. The reason for this decision is a different motive in terms of publishing content as part of research, rather than commercialising the product.

- **Developer/Creator:** The role of the developer or creator envelops a broad scope of tasks related to either releasing an AI system, advancing it, maintaining it, or multiple of these activities. More specifically stated, the scope includes every individual actively contributing towards developing, operating, extending, optimizing, and/or maintaining the AI system. Some detected examples of roles fulfilling this description would be *IT Engineers*, *IT Architects*, *Data Scientists (working within the scope of a development/operations team)*, *AI Engineers*, *Feature Engineers*, or *Designers.* One additional point to annotate is the boundary to the respective field used. The developer or creator does not have to be a subject matter expert in the field of usage of the AI system. One example would be an *Expert in bio-informatics.* While it certainly is possible that the developer also occupies this role, it is no pre-defined requirement for developing such systems [KT21], [KT22], [MHDSG23],[RRF$^+$22], [PHB$^+$18], [Gün20], [Mil22].

- **Subject Matter Expert:** This role is concerned with providing the necessary domain knowledge to the AI system to use. As indicated by the developers, the individual(s) may also be the developers simultaneously. However, if this is not the case, the subject matter expert usually provides insight in tasks like the decision-making processes, specific behaviour and routines in occurring scenarios when operating in a specific domain, preparations and precautions for risks, as well consulting on domain specific topics [GJS22], [KT21], [KT22], [MHDSG23], [RRF$^+$22], [PHB$^+$18].

  It should be noted that the role of the Subject Matter Expert can only be considered as a broad term that covers a variety of different roles in which they can manifest themselves. In the research conducted for this paper, the main areas identified were the use of AI in the medical and financial sectors. To provide further insight, a brief additional list of how the Subject Matter Expert may appear in real life scenarios is provided.

  - **Financial sector:** In the financial technology sector, subject matter experts have been described as either bankers in general [KT22], *consultants* or *advice professionals*, which are responsible for advising in the development and maintenance of an AI system in the respective field of application [HCHVD21].
  - **Medical sector:** In the case of the medical domain, subject experts were considered as practitioners using the help of AI system for diagnosis. Consequently, the expert could occupy the role of a *Physician*, *Specialist (E.g., Neurosurgery, or ENT physician*, or *Nurses* [BML$^+$23]. However, the role does not necessarily have to be an ”executing” role. One role was found, which is responsible for clinical programs, namely the *Clinical Operations Officer*, and thus has been considered a subject expert for one scenario [GJS22].

- **Product Owner:** The role of the product owner mainly focuses on the party making decisions during development and operation of an AI system [Mil22]. In practice, this role may be covered by more commonly known roles, such as a *CEO* or *CTO*. However, the Product Owner does not necessarily have to be one of these individuals. It may also be an individual or party specifically assigned to lead the development and/or operation of a specific AI system [KT21], [RRF$^+$22], [Gün20], [Mil22].
- **Shareholders/Investors:** Shareholders or investors designate the group of individuals which either financially support an endeavour using an AI system, or directly invest in an AI system. While there are many forms of investment possible, this stakeholder was rarely mentioned in literature. Consequently, the investors are considered granular enough for the scope of this work [Gün20], [RRF$^+$22].
- **Internal Audit** The stakeholder group associated with internal audits may be referred to as sort of ”governance” in accordance to commonly applicable standards and norms. Among the main interests of these auditors would be examples like ensuring that legal requirements are adhered to, fulfilling quality standards, ensure that other regulations required by sources, like service contracts are met, as well as checking if there are complementary assets, like a risk management concept, are at place. [KT21], [KT22], [MHDSG23]

4.1.3. User

User entail private users, as well as as other companies using an AI system offered by any other party described in the producer section 4.1.2. To elaborate on both sub-groups, a private user can be any user who has access to an AI system, as well as the ability to use the system (should any requirements, such as internet access or downloading a local version, be a prerequisite for using an AI system). If a company poses as the user on the other hand, the term ”user” may be considered as all parties within said company using the AI system to conduct their tasks. This would entail the ”general” role of a worker (e.g., using LLMs for a task) as well as specialists, like the previously mentioned *Subject Matter Experts.*

That being said, one important distinction is to be considered. Contrary to a Subject Matter Expert directly involved in the development or operation of an "intern" AI system, said experts have to work with a system externally produced and maintained. Thus, the ability to use and alter such system highly depends on the individual scenario. As a result, the requirements may shift drastically. However, for the scope of this work, general usage as provided by the producers is assumed [HMD+23], [Gün20], [WBD+21], [RRF+22], [PHB+18], [Mil22].

4.1.4. Academic Sector

For the distinction within this work, the academic sector includes all individuals involved in studying AI systems. Similar to the producers, this activity may include the development of AI system. However, it is no mandatory requirement. As such, the focus of the academic sector is set on understanding AI systems better as well as advancing the subject further. This distinction to production will be reflected more clearly in the specific roles in academia following shortly, as well as the context of the requirements of the academic sector.

- **AI researcher:** The role of the AI researcher pertains to any individual or group actively doing research on an topic at least including AI [PHB+18]. Said research may target at AI as abstract concept (i.e., neural networks which can be applied in various scenarios), or in specific domains, like the aforementioned medical field [GJS22], [PHB+18].
- **Professors:** Professors working on AI extend the more abstractly described role of an AI researcher by also teaching related concepts in addition [PHB+18], [PHB+18].
- **Students:** Students are any individuals who are taught by a professor in courses related to AI and AI systems [PHB+18], [PHB+18].

4.1.5. Indirectly Affected Individuals/"Bystander"

Bystanders include any person or party who is not directly involved in any activity related to an AI system, but who may be affected by the use of an AI system. Affection may manifest itself in a variety of ways and may or may not be directly recognisable by those affected. One prominent example would be the disclosure of information by disclosure of the training data used. But the result may also be any bot acting as agent interacting with a human being. Another example would yet again being a patient whose medical condition is assessed by an AI system to support the diagnosis process [GJS22], [WBD+21].

4.1.6. Ethic Activists

Ethic activists revolve around stakeholders which incentive and promote ethical usage of AI systems. This may concern topics like unintentionally harming other human beings by autonomous decision making, ethical concerns for autonomous execution, discrimination due to biased input data, misuse of such system by external parties, or negative impact on employees [PHB+18] [KT22].

4.2. Requirements

Shifting the attention away from the stakeholder, the focus may now be set on the requirements. The aforementioned literature review yielded insightful insights, which have revealed a rather complex underlying nature concerning the respective requirement. Multiple instances of synonymous definitions for the same result, as well as the inverse case have been detected. In more detail, in the first case requirements such as *"explainability"* and *"understandability"* refer to the same need in some scenarios. Contrary, some requirements found were denominated with the same term, but differed vastly in the underlying need, depending on the current context.

Consequently, the very same requirements are to be explained from multiple perspective. Thus, the following listing of requirements is structured as follows:

1. The requirement is elaborated in general.
2. While iterating through the different stakeholders introduced in 4.1, first the requirement is mapped to other terms used in a synonymous fashion, should such case occur.
3. Similarly, after showing the synonyms, potentially differentiating context will be elaborated.

In addition, the following list is by no means exhaustive, as the scope of this work is limited to requirements for the abstract concept of 'AI systems'. Concerning the structure of the enumeration, the detected requirements will be categorized into functional- and non-functional requirements. Functional requirements cover those directly applicable to the notation used, while non-functional requirements cover general quality aspects expected in the documentation process. One example of such a quality would be "usability" [Gli07]. However, the following list of identified functional requirements is based on a more detailed categorisation, as they cover a range of additional content of the general environment of the application of the AI system, data requirements, additional content and perspective of the workflow itself, as well as ethical considerations. As such, they will be categorised according to these general themes. Table 4.1 provides an overview and description of said categories.

Category	Shorthand	Description
Workflow Requirements	*WF*	Requirements concerning the documentation of the work-flow itself
Data Requirements	*DA*	Requirements for the use and management of data and data flows within and around the AI system
Ecosystem Requirements	*EC*	Requirements related to the domain and environment of the AI system application
Documentation Structure	*ST*	Requirements for the structure of the documentation itself
Ethical Requirements	*ET*	Requirements on how certain ethical concerns are managed

Table 4.1.: Categorisation of the Functional Requirements

4.2.1. Functional Requirements

First, the functional requirements are covered in order to show what type of content is expected to be included in the documentation. Structurally, the enumeration follows the categories and will be labeled with the shorthand of Table 4.1 followed by a sequential number.

4.2.1.1. Workflow Requirements

For he workflow itself, the following requirements have been found:

- **[WF1] Description of Design Decisions:** For the AI components, each design decision as well as a version of changes should be documented. In this case, design decision include data usually contained externally in configuration-files or diagrams, and which are easily adaptable, such as hyper-parameters, potentially the number of layers and hidden units, or comments. Furthermore, any necessary modifications of the aforementioned tools are supposed to be described in the documentation as well [KT21], [HMD$^+$23], [KHK$^+$24], [GJS22], [LWM24].
- **[WF2] Description of Purpose:** The purpose of the AI system should be described in detail. This is essentially a counterpoint to the description of limitations, as it should be clear what the AI system is intended to do and what behaviour and outcomes can be expected [HMD$^+$23].
- **[WF3] Description of Robustness:** Robustness refers to the level of confidence to which an expected outcome can actually be predicted. In terms of documentation, it should also be shown, how the system will not produce unexpected results in the case of new data being used to train the persisting AI system [KT22], [CDVR22], [WBD$^+$21] .

- **[WF4] Conclusions about Output:** For every system acting as advisory for decisions or delivering recommendations, it is paramount to understand how the output was calculated by the system, especially when unexpected outcomes are occurring. Furthermore, it is to be stated how the results were interpreted, if said results differ significantly from previous results [HMD+23] [PHB+18] [GJS22].

- **[WF5] Description of System Architecture:** In the documentation, the interacting components and their way of interaction are to be described in an understandable way for other developers as well as other parties having to potentially review and/or audit the system [HMD+23], [KHK+24], [WBD+21] [GJS22].

- **[WF6] Noting Measured Accuracy:** The average/median accuracy of the system should be known by stakeholders, as it is one of the main key performance indicators [HMD+23], [WBD+21] .

- **[WF7] Description of Training Process:** This requirement involves the training techniques, parameters for relevant functions, trade-offs, and assumptions used to train the models used within an AI system [HMD+23].

- **[WF8] Documentation of the Testing Process:** In particular, to demonstrate due diligence and quality assurance, the testing process and thus the evidence of working properly should be documented [HMD+23].

- **[WF9] Documentation of Changes and Change Management:** Similar to the management of different versions in software, change management and version history should be established and maintained. In more descriptive terms, a reviewer of the documentation should also be able to trace what previous versions of the system looked like, for example by looking at previous diagrams. In addition, for authorities, the robustness of the changes must also be described and it must be demonstrated that the system still maintains the quality and safety measures it had prior to the changes [HMD+23], [KHK+24].

- **[WF10] System Description:** In addition to the system architecture and other technical details revolving around what components are used to run the AI system and how these components may interact with each other, the environment such as hardware, operating system, data communication used by the AI system must also be described [HMD+23].

- **[WF11] Description of Scalability:** Especially for using an AI system commercially, it may be relevant to know details about how many customers can be supported and which limitations at spikes in demand are to be considered [PHB+18]

- **[WF12] Display of Limitations:** Known limitations of the system should be included in the documentation in a clear and understandable manner to provide the other stakeholders with the necessary knowledge for further decisions. Examples would be to show which needs of customers can be met, or to let users know that certain content can not be generated due to the law or other policies [MHDSG23].

4.2.1.2. Ecosystem Requirements

Concerning the documentation needs for the ecosystem the AI system operates in, the following requirements have been found:

- **[EC1] Display of Risk Management:** Especially when it comes to high risk AI systems, the documentation should include various points to effectively communicate what risks exist, and how the system mitigates or eliminates these risks [MHDSG23]. Risk management is also further defined by different literature to the extent that the exact points entailed depend on the point of view.

  From the users lens, risk management focuses on displaying how potential harm to humans is prevented. For instance, this may entail the usage of sensible data and how it is stored. It is therefore crucial to understand what risks exist and how the outcomes may affect these individuals [KT22], [MZM21], [WBD+21].

  For auditing and risk assessment, the view usually converges to a more technical framework. In addition to the transparency of the system and its technical details, risk management usually encompasses a large number of items which are also included in other parts of this list, such as *Robustness*, *Display of Incident Management*, *Change/Version Management*, *Dataset Scope*, *Data Preparation Process*, *Data Provenance*, and *Data Protection Measures* [EuA24c], [HMD+23], [MHDSG23]
- **[EC2] Display of Incident Management:** In certain cases, the documentation shall include a section containing a plan for potential threat scenarios and how to deal with them in the particular fashion and environment in which they can occur. An example of such an event might be adversarial attacks on an ML component used in a forecasting system.[HMD+23], [LWM24] .
- **[EC3] Description of the Application Domain:** Documentation should also reference the environment of application to allow for better comprehension of other requirements occurring in this list, such as transparency in general, design decisions, certain risk factors, or a description on outcomes [KT21].
- **[EC4] Description of Safety Processes:** If any internal safety processes are in place to prevent harm, they are to be included in the documentation. One instance of such an internal safety measure could be a mechanic to prevent accidental misuse by user [HMD+23]. Usually, such safety processes are delivered alongside the *risk management.*
- **[EC5] Display of Interaction Error Risk:** In the documentation, potential sources of errors during interacting with the AI system should be addressed. Furthermore, if such sources of errors are known, the developed and established mitigation measures are to be described. This could for instance be accomplished by describing how humans interact with the system, or by employing an element resembling a human actor in the system [HMD+23].
- **[EC6] Description of Human Roles in the System:** Within the documentation, each human actor being involved in the general ecosystem and environment of operation is

to be described. This measure helps to better understand how different stakeholders are influenced and/or affected by the AI system [HMD+23], [KT22].

- **[EC7] Modules for stakeholders:** There should be different components of the documentation tailored towards different stakeholders. Furthermore, the components may be designed to consider different levels of abstraction and sensitive information to protect said sensitive information [MHDSG23], [HHM+20].
- **[EC8] Data Privacy:** Data privacy was usually used in literature in two different perspectives. Generally described for the scope of this work, data privacy revolves around preventing the disclosure of information. For any individual user, the answer is usually straight forward. No sensitive information about said individual is to be published either willingly or unwillingly [BAW+20], [GJS22], [KT21].
- **[EC9] Inclusion of Accountability:** The documentation should display the roles in an organization using AI systems held accountable in case of erroneous behaviour or any other damages occurring [CDVR22] [MHDSG23] [CDVR22].
- **[EC10] Quality Management:** The documentation should describe quality measures introduced to the AI systems, as well as repercussions of these quality measures. An example would be a pre-screening of a response to filter out unwanted content from any generated text sent to the user [EuA24a].

4.2.1.3. Data Requirements

Particularly for managing the data the AI system uses and interacts with, numerous requirements have been found. Said needs encompass:

- **[DA1] Description of the Training Data:** In the documentation, a general description of the training data should be embedded. Depending on the literature reviewed, said description may entail the *data origin* [KNHJ+23], [HMV+22], the *data collection process* [KNHJ+23], *the data preparation* including the *data cleaning process* [HMD+23], the *filtering process* [KNHJ+23] with the inclusion and exclusion criteria (if any applied), and other design decisions such as any *artificial data enrichment* used [KNHJ+23] [KT21]. Any of the data manipulation methods above are to be emphasised at this point, as they are usually not routinely documented with the level of detail required to conclude meaningful reasoning behind certain behaviours of an AI system [KNHJ+23]. Furthermore, depending on the position on data privacy, additional transparency in form of full disclosure of the training data is also advocated [MHDSG23].
- **[DA2] Data Provenance:** The origin of the data, as well as any external references are to be elaborated as part of the documentation of the data management [HMD+23], [KNHJ+23].
- **[DA3] Description of the Data Collection Process:** For the data documentation, the process of collecting or obtaining the data should be included [HMD+23], [HMV+22] [GJS22].

- **[DA4] Description of the Data Preparation Process:** Any manipulation of the data before it is used in the system should be traceable to promote transparency and explainability [HMD+23].
- **[DA5] Report on Data Relevance:** Proof of data relevance for the use of the AI system may be required for risk labelling and authorisation by audits. Furthermore, documenting relevance helps to further elaborate how and why certain data is used by the AI system [HMD+23], [KNHJ+23].
- **[DA6] Inclusion of found Dataset Anomalies:** If any anomalies are found in the data to be used in the AI system, these should be reported and the rationale for the use of these anomalies should be described [KT22].
- **[DA7] Documentation on Bias and Fairness:** Documentation of the data should include a section on any biases identified and measures taken to correct them [CDVR22], [WBD+21], [DYMY21], [LWM24].
- **[DA8] Display Data Validation Measures:** It should be documented if there are any specifics how the data and the results have been validated, as well as between the logic between the problem formulation and the result [HMD+23], [GJS22].

4.2.1.4. Documentation Structure

The following documentation structuring needs were identified

- **[ST1] Documentation Standards:** The documentation of AI systems should follow documentation standards pre se. However, no such standards were found, which are currently being established [KT22], [CC22], [HMD+23], [LWM24].
- **[ST2] Clear Documentation Guidelines:** Documentation in general should follow documentation guidelines, if any are applicable. However, as the subject of AI systems is a comparatively new field of research, often the guidelines used are known sources of influence, such as the *EU AI Act* [EuA24a]. However, as far as the author is aware, there is no documentation guideline in place at the time of writing, so the challenge may be to adapt to future mandatory documentation requirements [KT22], [CC22].
- **[ST3] Raise of Incentive:** One problem that tends to be prevalent in software engineering in general is that documentation is seen as a low priority, and thus there is little incentive to write (extensive) documentation [CC22]. Consequently, one requirement would be to achieve a form of documentation that is seen as ”less of a chore”.

4.2.1.5. Ethics Guide

[ET1] Documentation for AI should act as an ethics guide and as such contain explanations on how certain decisions reflect the ethic rationale, acknowledge what potential harm it could induce to its environment of operation and how these harms are mitigated or eliminated [BHKST22], [MHDSG23],[MZM21], [DYMY21], [PHB+18], [KT22], [LWM24].

4.2.2. Non-Functional Requirements

Following on from the functional requirements for features, the upcoming list covers the requirements, not linked to ”tangible aspects”, that are expected when producing documentation and working with a notation.

- **Transparency:** Starting with the most commonly used term, transparency in the context of AI systems is generally described as any party examining a system also being able to understand what the concept of that system is and why it produces certain results based on the underlying concepts. However, as it ways pointed out by literature, transparency has many facets which are highly dependent on the *individual or party using the documentation*, and *the level of detail* which is currently of interest [MHDSG23], [WBD+21], [KNHJ+23], [PZK22], [LV23]. Consequently, there are multiple layers to consider for transparency.

  The first category to consider would be the target audience. Depending on who is reviewing the documentation, the meaning of transparency could shift from the high-level description stated above to full disclosure of the source code alongside the documentation. This need would be raised, for example, by legal auditors tasked with assessing the risk potential of AI systems.[BHKST22], [WBD+21]. But the scope of discrepancy is not limited to the description, but to other characteristics as well. For instance, for the users a characteristic of transparency would be to clearly identify AI as such and not confuse it with another human [WBD+21]. For accidents reviews or quality assurance, transparency especially refers to showing decision points [LWM24], [WBD+21]. Considering that parties with potentially deeper knowledge of such systems, such as subject matter experts, can demonstrate quality management in general, as well as how results are reproducible, is considered part of transparency [KHK+24],[WBD+21].

  Focusing more on the level of detail mentioned above, when transparency is used as an abstract term, the ability to show what an AI system is doing and why certain outcomes occur often seems to be the correct definition. [BML+23], [CDVR22], [KNHJ+23], [MHDSG23]. That being said, various papers have been found, which are using transparency to describe a certain part of a system. These detailed descriptions entail the *description of data-sources* [HMD+23] or the , *description of technical details* [EuA24c], [MHDSG23]. Finally, one small annotation would be that generally, the term *explainability* is also used to describe the characteristics pointed out for transparency [WBD+21].

- **Unambiguity:** In the context of documentation, unambiguity is broadly described as the ability of each person reviewing the documentation to gain exactly the same knowledge. In other words, the documentation should leave no room for interpretation, regardless of who is reviewing it [CC22], [CDVR22], [KT21], [KT22], [PHB+18], [RRF+22]. Described in other words, the documentation should act as a *Communication Artifact* [KT21].

In practice, ambiguity in (AI) documentation was often described as descriptions only understandable by people having knowledge in programming or the usage of symbolic data representation [PHB+18], [CC22]. So part of the requirement would be to consider non-experts and non-programmers for the documentation, or at least some modules of it, tailored to the diverging needs of non-experts and non-programmers.

- **Granularity:** Granularity generally refers to the challenge of choosing the right level of detail without the documentation getting cumbersome. More directly described, the documentation should be adapted aptly to the other requirements without loosing any information. For instance, if the AI system is considered non-risky and technically ”easy” to comprehend, then a documentation explaining the purpose and the results might be sufficient. However, in practice the decision often may not be so easy [CC22], [CDVR22], [KT21], [HCHVD21].

- **Reproducibility:** Reproducibility in the context of AI systems describes the capabilities to set up a test system of similar structure to the original system and using the same input data as well as settings to check if the same output can be generated [KT22],[KHK+24], [MHDSG23]. Another aspect to consider is the resource usage. Similar to *scalability*, other researcher ideally should have the option to test AI systems also at smaller scale. Furthermore, the entire process should be accomplishable by using the documentation only [KT21]. Finally, emphasis should be placed on keeping as much data as possible in the original training state, so that as few confounding factors as possible interfere with the reproduction of the original system [KNHJ+23].

- **Low Learning Cost:** Documentation should act as aiding tool, rather than a hindrance. Thus, people who have never seen the documentation before, should be able to follow the documentation without too much effort of learning. This circumstance was also described as *Data Artifact* [KT21]. To achieve this goal, further steps can be taken, such as translating technical jargon into non-technical descriptions, legends or other auditory tools that describe the work, rather than just ”plain” stats and facts [CC22], [PHB+18].

- **Explainability:** Explainability generally refers to the same concept as some definitions of *transparency*, and thus aims at an understanding of the reading party with an focus on decision making mechanisms. However, there is one distinction to be made from transparency. Namely, explainability usually focuses on different reviewers having similar levels of knowledge about the AI system. Consequently, by this definition, non-experts are excluded [WBD+21], [KNHJ+23] [PHB+18].

  While the same principal is usually true for businesses as well, there are some annotations to be made. First, training data may contain some degree of sensitive information from the individuals said data originates from. As established in the section about *transparency*, audits commonly demand full disclosure of the entire system, which also encompasses training- and test-data as well. Thus, the option to show encrypted data may be required legally and a conflict of interest for using such data may be at risk. However, it is not within the scope to delve into this topic further. For other parties, such as users, the statement from the first paragraph still applies [HMD+23], [KT21].

Second, the degree of explaining data privacy measures varies per stakeholder. For users, the general explanation of measures to protect the data may be sufficient. That being said, it was also stated that users should be able to verify claims on data privacy [BAW+20]. Nevertheless, for auditions, the description of safeguards implemented, the rationale of sufficiency, and other technical details would have to be included in the documentation as well [GJS22], [KT21].

- **Adaptability/Flexibility:** Documentation should be flexible, extensible, and modular to be receptive for potential changes required in the future [PZK22], [KT21]. However, a different characteristic fits the description of adaptability. Specifically concerning diagrams, the challenge to adapt existing diagrams to the requirements provided to sufficiently has been mentioned. Thus, if a documentation tool is used, various pre-defined templates would be beneficial [CC22], [GJS22].

4.3. Overview - Stakeholder and Requirements

To conclude the first section of the results, a broad overview of the stakeholder and their respective requirements will be visualised in the Tables 4.2 for the functional requirements, as well as the Table 4.3 for the non-functional requirements.

Stakeholder Group	**Category**	**Requirement**	**Reference(s)**
Legal Entities	**Workflow Requirements**	[WF1] Description of Design Decisions	[HMD+23] [KT21] [MHDSG23] [WBD+21]
		[WF2] Description of Purpose:	[HMD+23] [MHDSG23]
		[WF3] Description of Robustness	[HMD+23], [KT22] [WBD+21]
		[WF4] Conclusions about Output	[HMD+23]
		[WF5] Description of System Architecture	[HMD+23]
		[WF6] Noting Measured Accuracy	[HMD+23] [WBD+21]
		[WF7] Description of the Training Process	[HMD+23]
		[WF8] Documentation of the Testing Process	[HMD+23]
		[WF9] Documentation of Changes and Change Management	[HMD+23]
		[WF10] System Description	[HMD+23] [WBD+21]
		[WF12] Display of Limitations	[KT21] [MHDSG23]

	Ecosystem Requirements	[EC1] Display Risk Management	[EuA24c] [BAW+20] [HMD+23] [KT21] [MZM21]
		[EC2] Display of Incident Management	[HMD+23]
		[EC3] Description of the Application Domain	[KT21]
		[EC4] Description of Safety Processes	[KT21]
		[EC5] Display of Interaction Error Risk	[HMD+23]
		[EC6] Description of Human Roles in the System	[HMD+23] [KT22]
		[EC7] Modules for stakeholders	[HHM+20]
		[EC8] Data Privacy	[KT21] [BAW+20] [HMD+23] [MZM21]
		[EC9] Inclusion of Accountability	[MHDSG23]
		[EC10] Quality Management	[EuA24a]
	Data Requirements	[DA1] Description of the Training Data	[HMD+23] [MHDSG23]
		[DA2] Data Provenance	[HMD+23] [MHDSG23]
		[DA3] Description of the Data Collection Process	[HMD+23] [MHDSG23]
		[DA4] Description of the Data Preparation Process	[HMD+23]
		[DA5] Report on Data Relevance	[HMD+23] [MHDSG23]
		[DA7] Documentation on Bias and Fairness	[WBD+21]
	Documentation Structure	[ST1] Documentation Standards	[BAW+20]
	Ethical Requirements	[EC1] Documentation as Ethics Guide	[MZM21] [MHDSG23] [BHKST22]
Producers	**Workflow Requirements**	[WF1] Description of Design Decisions	[GJS22]
		[WF4] Conclusions about Output	[GJS22]
		[WF5] Description of System Architecture	[GJS22]
		[WF6] Noting measured Accuracy	[GJS22]
	Ecosystem Requirements	[EC1] Display Risk Management	[MZM21]

		[EC8] Data Privacy	[MZM21] [CDVR22] [GJS22]
		[EC9] Inclusion of Accountability	[CDVR22]
	Data Requirements	[DA1] Description of the Training Data	[KNHJ+23]
		[DA2] Data Provenance	[KNHJ+23]
		[DA3] Description of the Data Collection Process	[GJS22]
		[DA5] Report on Data Relevance	[KNHJ+23]
	Documentation Structure	[ST3] Raise of Incentive	[CC22]
	Ethics Guide	[EC1] Documentation as Ethics Guide	[MZM21] [BHKST22]
Academic Sector	**Workflow Requirements**	[WF1] Description of Design Decisions	[KHK+24] [LWM24]
		[WF4] Conclusions about Output	[LWM24]
		[WF5] Description of System Architecture	[KHK+24]
		[WF9] Documentation of Changes and Change Management	[KHK+24]
	Ecosystem Requirements	[EC2] Display of Incident Management	[LWM24]
		[EC9] Inclusion of Accountability	[BML+23]
	Data Requirements	[DA1] Description of the Training Data	[HMV+22]
		[DA3] Description of the Data Collection Process	[HMV+22] [LWM24]
		[DA7] Documentation on Bias and Fairness	[BML+23] [DYMY21]
	Documentation Structure	[ST1] Documentation Standards	[LV23]
		[ST2] Clear Documentation Guidelines	[KHK+24]
	Ethics Guide	[EC1] Documentation as Ethics Guide	[KHK+24] [DYMY21]
User	**Workflow Requirements**	[WF2] Description of Purpose	[HMD+23]
		[WF3] Description of Robustness	[HMD+23]
		[WF4] Conclusions about Output	[GJS22]
		[WF6] Noting measured Accuracy	[HMD+23]

		[WF9] Documentation of Changes and Change Management	[HMD+23]
	Ecosystem Requirements	[EC1] Display Risk Management [EC2] Display of Incident Management [EC5] Display of Interaction Error Risk [EC6] Description of Human Roles in the System [EC8] Data Privacy	[HMD+23], [KT22] [HMD+23] [KT22] [KT22] [GJS22]
	Data Requirements	[DA1] Description of the Training Data [DA3] Description of the Data Collection Process [DA4] Description of the Data Preparation Process [DA6] Inclusion of found Dataset Anomalies	[HMD+23] [HMD+23] [HMD+23] [KT22]
	Documentation Structure	[ST1] Documentation Standards [ST2] Clear Documentation Guidelines	[KT22] [KT22]
	Ethics Guide	[EC1] Documentation as Ethics Guide	[KT22] [BHKST22]
Ethic Activists	**Ethics Guide**	[EC1] Documentation as Ethics Guide	[KT21]

Table 4.2.: Stakeholder & Functional Requirements

Stakeholder Group	**Requirement**	**Reference(s)**
Legal Entities	Transparency	[KT21], [MHDSG23], [HMD+23] [KR21] [MHDSG23] [WBD+21]
	Unambiguity	[KT21], [HMD+23]
	Low Learning Cost	[KT21]
	Adaptability	[KT21]
	Reproducibility	[MHDSG23]
	Explainability	[WBD+21]
Producers		

	Transparency	[PHB+18] [KNHJ+23] [KR21] [CDVR22] [CC22]
	Unambiguity	[PHB+18] [CDVR22] [CC22]
	Scalability	[PHB+18]
	Explainability	[PHB+18] [KNHJ+23] [CDVR22]
	Reproducibility	[KNHJ+23]
	Granularity	[CDVR22] [CC22]
	Adaptability	[PZK22] [CC22] [HMV+22] [GJS22]
Academic Sector	Transparency	[KHK+24] [BML+23] [LV23] [HCHVD21] [LWM24]
	Reproducibility	[BAW+20] [KHK+24]
	Explainability	[PHB+18] [BML+23]
	Adaptability	[PZK22] [HMV+22]
	Low Learning Cost	[PHB+18]
User	Transparency	[PHB+18] [LV23] [WBD+21]
	Unambiguity	[PHB+18] [RRF+22]
	Scalability	[PHB+18]
	Explainability	[PHB+18]
	Reproducibility	[KT22]
Bystander	Transparency	[WBD+21]

Table 4.3.: Stakeholder & Non-Functional Requirements

5. AI System Documentation Notation

With the stakeholder and requirements covered, the next step would be to find and compare different techniques used to document AI systems. As AI systems have so far been predominantly used to document in either textual or visual format, many common approaches to document software in general have been found. This chapter will focus on providing a description on the found techniques, and how well they would likely fair against the found requirements.

To do so, various different visualization and knowledge representation- and storage methods have been compared against each other. The assessment is based on how well the established requirements are met. Before starting with the chapter, the structure of the chapter is to be explained. While only notations that have actually been used to document AI are included, the approaches found can vary widely in purpose, nature, and the overall level at which the system is represented. For example, some of the used methods follow a general purpose modelling notation suitable for different types of models. Contrary, some of the techniques are specifically designed to represent AI systems. Finally, there are some techniques which do not fit any of these categories sufficiently and can be considered more of a specialised approach. Thus, the chapter is separated into found *modelling languages* 5.1, *AI-Specific notations* 5.2 and *specialised approaches* 5.3.

Additionally, it should be annotated, that the different notations are naturally not designed to cover all the requirements. However, to outline how different notations may complement each other, each representation is compared against all categories of requirements regardless. Furthermore, requirements regarding the documentation structure are not covered, as these requirements affect the whole documentation, rather than a single diagram. The chapter then concludes with a summary of the advantages and disadvantages of using one model versus multiple models, while also discussing the option of complementary models.

5.1. General Purpose Modelling Languages

The first section on notations covers the most common approach, which is to use already known methods to represent concepts in software. Furthermore, some of the emerging techniques have been in use for a couple of decades and have thus gone through numerous steps of adaptation and optimisation [vdML02]. As such, these approaches also have been adopted for the documentation of AI systems. In the following, the suitability for the representation of AI systems in particular will be discussed.

5.1.1. UML

One ubiquitous contestant when it comes to display and relay information is the Unified Modeling Language (or UML). As the name may suggest, UML is a modelling language designed for documenting and visualizing various technical concepts. The language comprises of different sub-genres of models and model types, which all have their own individual characteristics to consider. Based on these prerequisites, it is assumable that almost every requirement can be met to some extent by a combination of multiple UML diagrams. Nevertheless, there are some issues to consider, which are true for UML generally, as well as for some of the specific diagram types designed in the UML notation [MLNN20], [HG22], [JPZ22], [AF21].

In order to prevent redundancy in the information, the next sections will covering different aspects of UML. First, the advantages and disadvantages of UML in general will be elaborated. Following the general elaboration, the diagram types considered most relevant for the requirements will be introduced briefly. Furthermore, the their benefits and drawbacks not already covered in the general section will be outlined.

Starting with the benefits of UML in general, the notation is a well established standardised form, which is used vastly in the software industry as well as in the research. As such, many people are familiar with at least a portion of UML diagrams and are able to read other types of UML diagrams as well. Furthermore, UML is rather static in nature, meaning that usually each element in a diagram has definite meaning. This ensures that diagrams of the same type can be read in every instance, regardless of the current scenario said model is applied to. Finally, it should also be mentioned, that elements used in multiple diagram types keep their assigned purpose per default. For instance, the symbol representing an actor used in an interaction diagram also represents an actor in use case diagram. As such, UML diagrams are generally good candidates for most workflow requirements, ecosystem requirements, and data requirements [AF21], [HG22].

With the advantages covered, the problems when using UML to draw AI systems are also to be highlighted. Standardisation in general pictures the inverse of flexibility. Because UML elements have a predefined meaning, there are some scenarios where certain diagram types do not have the necessary granularity when using the elements provided. For instance, flow or activity diagrams are a suiting option to represent the general workflow. However, it would be challenging to include more information on individual machine learning components, if required. Moving on, especially for larger systems, some diagram types tend to become very cluttered, as many details could be omitted, depending on the level of abstraction and context. Regarding comprehension, the UML language is considered to have a higher cost of learning for parties who have not worked with UML diagrams beforehand, as the notation is based predominantly on concepts found in software development. Finally, it generally is not envisioned to attach arbitrary additional information, such as further details [MLNN20], [HG22]. As such, individual UML diagrams would have to rely on external sources, if textual content like ethical guidelines are to be included in the diagram as well.

UML - State Machines

State machines in general depict a software system in its different stages of operation. With the representation options provided, they allow to efficiently depict either a portion of a system once a "relevant" step has been concluded, or how exceptions are handled, should any occur [Rum16], [KM02].

As such, they would pose as fitting option to represent some sections of the workflow of an AI system as well. For instance, it could be used to cover the process of the data collection and the data triage process. Another example would be to illustrate implemented safety protocols and how they apply in different scenarios. Thus, they would be able cover many data and ecosystem requirements [Rum16], [KM02].

That being said, there are some aspects to consider when using state machines to represent the workflow. Firstly, in most state diagrams, the transfer of information is highlighted. However, as the requirements highlight, technical details ought to be displayed in addition. UML state machines do not contain too many technical details, or some workflow details only in an abstract way. As such, they would require complementary diagrams to include the missing information relevant for documenting AI system in terms of the found requirements [Rum16], [KM02].

UML - Use Case Diagrams

Use case diagrams display the interaction with a software system with different actors. The actors in this case can be humanoid, or non-humanoid (for example machines, sensors, or other software components, like a listener in a broadcasting system) [vdML02]. In a standard scenario, use case diagrams are modeled in a more abstract fashion, but do have the option to display more details about the interactions with the system.

Keeping the last piece of information in mind, use case diagrams present a well fitting opportunity to model and elaborate how such actors may cause errors in the system [SL04]. Even if not focused on errors, use case diagrams can be used to show how users experience the interaction with the system and may display critical states of the system to take special care of. One example may be handling data privacy of users in form of displaying all options a user may interact with a system. As such, use case diagrams help to display various ecosystem requirements as well boosting transparency and explainability. Moreover, use case diagrams can be plotted in multiple ways and with multiple extensions. One popular example would be the use case meta-model. With such options, granularity and scalability can be ensured to some extent [vdML02].

Like all models, use case diagrams also have their share of disadvantages. Especially one factor is to be highlighted. Use case diagrams usually do not feature any technical details about the structure of the system and only display the workflow in terms of the interaction with the system, but not with the parts conducted automatically. As such, all the additional explanatory content highlighted in the requirements, such as system architecture, output description, data sources, risk management are not covered by use case diagrams alone. Thus, additional diagrams or textual descriptions would be required yet again [SL04], [vdML02].

UML - Activity Diagrams/BPMN

Activity diagrams in general may be considered the most obvious solution for displaying AI systems. Their main purpose is to depict a sequence of activities to accomplish a certain task. The diagram can be created in a number of different notations, some of which are outside the UML family. Two of the most prominent examples include general abstract Flow-Diagrams, or a specification thereof called *Business Process Model and Notation* (BPMN for short) [CT12]. However, given the roughly congruent objective of these notations, the focus for the scope of this work remains on outlining activity diagrams.

Activity diagrams consist primarily of the activities, which describe the actions taken at a particular step of the process to progress towards the final goal, and gates exercising different controls on the flow of activities. These gates may alter the flow by starting multiple activities simultaneously, looping the same task, or reverting some activities in case of an exception. Additionally, tasks can be assigned to specific roles. Said roles are shown by containers surrounding the workflow, where each task conducted by a specific role is placed within these containers. Finally, it is also to be annotated, that this type of diagrams usually also offers the option to nest an activity diagram within an activity, which adds more detail to the execution of the ”main” activity. Thus, the option of covering the process at different levels of granularity is also provided [KEBP21], [DTH01].

At this point it may seem logical to choose activity diagrams as means to document AI systems. Due to the purpose of this specific notation, transparency of the actual workflow is to be highlighted in particular. The ability to represent workflows, specific alternatives and exceptions that may occur during the use of an AI system, as well as the inclusion of multiple levels of granularity, are mentioned in the requirements in various forms. Furthermore, the process is not bound to describe only one dimension within the same diagram. It is possible to include the data collection and preparation process in the same diagram. Transparency and granularity can be maintained through sub-processes. Finally, BPMN also allows to attach notes and other standard artifacts. Thus, most workflow requirements and ecosystem requirements can be met to some extent [CT12], [KEBP21], [DTH01].

However, while it is possible to include details about the activities themselves, it is not the default to link information of other nature to activity diagrams, apart from comments. For instance, adding descriptions, such as describing the purpose of roles used as host of activities, is not an option in the default notation and would have to be linked to an external source. In addition, depending on the level of granularity, covering data requirements, such as describing the data origin, or ethical guidelines, may need to be included in the activities via workarounds. Depending on the scenario and the convention used to represent that scenario, activity diagrams thus can quickly become complex and overwhelming. This would negate the benefits described above [KEBP21], [DTH01].

UML - System Sequence Diagram

System sequence diagrams are similar to use case diagrams. They operate by displaying the interaction between users and objects. Contrary to use case diagrams however, system sequence diagram focus on the information flow. As such, they indicate what information is send between users and objects, where a user or an object can send information to other users and objects. They also incorporate an asynchronous component, as the delay to a response is also shown. For example, when a user submits a task to an AI system, a system sequence diagram can show how the request is passed to another component, which in turn may request information from a website, wait for the response, process said response, and return the result to the user who submitted the request originally [BHK+24].

Reviewing AI systems from this distinct point of view bears some interesting advantages. Message streams can display the data and in what way it would be modified throughout processing the AI system and as such is a great option to meet data requirements. Doing so would have the advantage of including various measures, such as showing in which sequences data is transferred encrypted [BHK+24].

That being said, given the size AI systems can scale up to, as well as different sections may have tasks conducted simultaneously, information streams can quickly get tricky to follow. Furthermore, deciding on the right granularity to display all data correctly and comprehensibly pose challenging. Finally, system sequence diagrams usually only have limited elements to plot. Thus, the system may not be expressive enough to display the variety of existing AI systems in terms of workflow or the surrounding ecosystem, as well as ethical guidelines [BHK+24].

UML - Class Diagrams

Class diagrams are a staple of object-oriented programming because they show different objects, their attributes and how they interact with other objects or themselves. They have various ways of expressing associations and realisations at different levels of granularity. It is also one of the most widely used notations within the whole family of UML notations [Rum16].

As such, modelling an AI system in form of a class diagram would be a logical option to tackle some of the requirements. Contrary to many other options before, class diagrams incorporate a wide variety of elements. For instance, there is a general "object", which would allow to tie specific content directly to other sources. Furthermore, comments are regularly used in class diagrams. Should the existing standardized options not be sufficient, there are numerous extensions for class diagrams. This variety is also conveyed to cater for granularity. Per default, class diagrams offer ways to attach details to generalized objects. Setting this statement into the context of this work, a general class "Machine Learning Module" could be created, and different implementations attached as a realization of this abstract base class, which inherently boosts granularity. As such, especially ecosystem requirements and workflow requirements can be tackled [Rum16].

However, providing a wider range of items to model with is bound to increase the cost of learning. Especially if a non-standard extension is used, even experienced users may need so time to familiarize with diagrams using said extension. Moreover, it will be challenging for many instances to follow the exact logical flow of execution steps of some AI systems, as only the association between objects, but not their order of execution is displayed typically. This extends to be a challenge for most of the data requirements as well, as the origin and processing can only be displayed in associations. Consequently, while it may be good to prevent ambiguity to some degree, transparency may suffer [Rum16].

5.1.2. TM Modeling

Another method which is greatly inspired by UML diagrams for displaying information is called TM modeling. TM modeling may even be viewed as an extension to UML diagrams, as it attempts to provide a transition from UML diagrams into the TM notation.It also provides the ability to combine multiple UML diagrams of different types (for example, a class diagram and a use case diagram) that represent a related concept and merge them into a single diagram. Furthermore, it can also feature layered perspective allowing to view different levels of granularity. To simplify the merge, other models are often simplified, thus creating these different levels of granularity during said simplification process [AF21], [KPL+14].

With these characteristics, TM models may be the best example to highlight all the benefits and drawbacks of most UML models. They allow to display vast quantities of information about complex concepts into comparatively little space and as such allow to include the workflow as well as the surrounding ecosystem, as well as other needed content. This benefit is further amplified by combining multiple perspectives into one meta-model. In addition, said snapshot of the scenario to document can also be achieved by uniform, distinct elements. As for data flows, the situation is similar to activity diagrams. Some information flows can be covered, but it is no focus [AF21].

However, maybe even more so than every UML diagram, these diagrams can get too complex rather rapidly. Furthermore, fully understanding the model correctly may take considerably longer compared to other options presented. Finally, learning to read the elements correctly can be a challenge in itself, as the same elements can have multiple meanings, depending on the context it is used in the diagram [AF21].

5.1.3. Ontologies

Ontologies are essentially a graph-representation of knowledge in an arbitrary degree of granularity covering usually either one or multiple related fields of knowledge. These fields of knowledge can be determined per use case and there are no limits set to the content. As such, depending on the actual scenario, an ontology can be as small or as grand as required. Furthermore, while the graph usually has a rule-set for syntax, such as setting up triplets or quadruplets, there is also little to consider regarding which content is stored in each node. That being said, considering Ontologies are graphs, even this rules may be adapted. For instance, if deemed necessary, a limit on the overall connections per node could be set as an additional rule [NMEC21], [SRMP+22], [DG23], [PHB+18], [KPL+14], [YZT+23].

This high degree of flexibility facilitates multiple striking advantages. Given that the scope of this work revolves around documenting AI systems, using an ontology easily provides the option to document such systems. In fact, it has already been done on multiple accounts [ELS+23a]. The style and structure of constructing the ontology can be tailored towards the individual requirements for each scenario it is applied. For instance, the basis of the ontology could be the workflow of an AI system. There could be a link to the individual components containing a description of said component and maybe further information on the functionality. The functionality may manifests in describing how the data is prepared, for instance. However, if just the workflow itself is not suitable, and further description on other aspects of the system, such as quality assurance, limitations, risk management, or documentation on bias and according countermeasures are required, it would certainly be possible to link these contents as well. As these examples illustrate, many of the requirements detected can be met. Ontologies are versatile, scalable and the flexible nature allows to essentially store each content required [DG23]. Finally, it should also be mentioned, that Ontologies are already recommended as format for displaying AI systems by some public policies of the EU and the W3C consortium [SRMP$^+$22].

While these qualities undoubtedly make ontologies a suitable candidate for documenting AI systems, most of these characteristics also have some drawbacks that need to be considered. Starting with storing arbitrary content, standardisation across different ontologies can only be achieved by convention and thus will strike as a challenge. Additionally, depending on the setup, the given flexibility and resulting variety in scale may render comprehension of the graph more challenging. As a result, boosting comprehension and minimizing ambiguity can only be prevented by establishing conventions on concepts like the level of granularity and the content featured in a node [NMEC21], [SRMP$^+$22], [DG23].

5.2. AI-Specific Notations

Apart from general modelling options, which have been used to display AI systems, other approaches have focused on AI solely. In particular, one type of boxology has been developed to more distinctively represent commonly utilized elements within an AI system.

5.2.1. Boxology

Typically, boxologies aim to provide a simple way of modelling different types of workflows. As the name may suggest, the central object is a rectangle, which represent crucial elements, such as data in the context of AI systems [VHTT19]. Additionally, there are other shapes to represent concepts like actors, activities, other objects which process data, and auxiliary sources, such as knowledge graphs, and a form of container marking a system, or a distinct component thereof [VHTT19], [ELS$^+$23a].

The simple nature of boxology leads to the core advantage of mostly intuitively comprehensive diagrams, even at larger scale. Though the level may be abstract, different components of a system can be read easily and the interaction between systems can be depicted usually uncomplicated by relaying the data transferred. Furthermore, with the elements provided, many workflow and ecosystem requirements can be covered. Concerning the data requirements, specific actions are covered via the processing element. However, other needs, like describing the data origin, are not included as default option [VHTT19].

This described simplicity also takes its toll in other ways. Firstly, incorporating different levels of granularity could be done by employing different components of the system at different levels (for example a similar concept to class-instance relationship originating from object oriented programming can be introduced in a diagram). However, said relation may quickly lead to confusion. Furthermore, while the introduced version of elements for boxologies is already tailored to represent AI system workflows, the elements may not be expressive enough to cover all aspects of an AI system. Finally, currently there is no easy way to feature other ecosystem concepts, such as risk management, easily into the workflow (at least without stretching the meaning of existing elements) [VHTT19].

5.3. Specialised Approaches

Finally, there are some approaches that aim to represent AI systems in particular, but not necessarily by providing a standardised form of notation. Rather, they ”embrace” the flexible nature of the matter and either provide options for representing the systems as desired in each specific case, or even eliminate the need for manual documentation of the system at altogether (though the latter option is aimed at developers and researchers who are specifically able to work with programming languages).

5.3.1. Data Cards

Data cards differ from all methods presented thus far. Contrary to all other methods, data cards are technically a textual description of the content to document. While visuals can certainly be incorporated, data cards are a structured summary capturing the essentials in context. To accomplish this task, the map itself is often divided into different sections covering the topic of interest, benchmarks and a brief description, although the scheme can be adapted as needed [PZK22].

These characteristics also mark the most distinguishing features. Having a summary in this fashion allows to convey larger quantities of information comparatively fast instead of having to read through paragraphs and enumerations. As such, they would be great to cover additional concepts like risk management, accountability, or descriptions of the training data. In addition, there are no strict limitations on how a data card can be structured, meaning that the content, overlay and size of the data card can be tailored to the specific application [PZK22].

However, flexibility comes again with the price of having to account for potential comprehension difficulties and standardisation by convention. Consequently, these requirements are entirely dependent on the party producing the data cards and may even be difficult to achieve. For example, while it may be argued that a clear level of granularity can be set, the decision to include or exclude certain details still leaves room for ambiguity and error, especially as this is a written format rather than elements in a diagram with a pre-defined purpose [PZK22].

5.3.2. Themisto

Themisto essentially is an automatic documentation generator aiding data scientists to document Jupyter Notebooks automatically via a plugin mechanic. To manage this task, the abstract syntax tree of the provided code is extracted and mostly matched against stored descriptions. In addition, there is also the option to manually prompt what the code is supposed to do. Based on the prompt, an answer based on online references will be generated by an AI model in the background. Based on the found matches, the most suiting responses are prompted back to the user. Themisto has primarily been tested on Notebooks published on *Kaggle* [WWD+22].

The authors present an intriguing approach with Themisto. Moreover, said approach covers a sizable number of requirements detected in the precious chapter. Especially each requirement aimed at describing individual parts of an AI system in a more detailed fashion may be accomplishable with Themisto. For example, one task often overlooked is to describe the data preparation process. This step could be conducted automatically. Thus, if working at a satisfactory level for the user, Themisto allows to reduce the task completion time of documentation. Furthermore, as already pre-defined answers are the default option, if applicable, using an automatically generated documentation also helps to provide a uniform documentation standard. For example, requirements such as transparency or accountability could be met by documenting the written tests and providing an overall system description [WWD+22].

That being said, there are a some limitations to be considered as well. Firstly, the purpose of the tool is to describe code. While this naturally entails AI systems as well, it is tailored towards individuals familiar with reading code. Thus, for the use case of this work, the majority of the stakeholder groups and by extension their requirements would be excluded to some extent. However, it is essential for AI systems that non-subject matter experts have the capability to understand how the AI systems operate, as they have to make decisions involving such AI systems. For instance, not every CEO tasked with making a decision on usage will have sufficient understanding by looking at code documentation.

But even for individuals adept in reading code, not all content is covered. Requirements such as describing data sources or data relevance, showing human interactions with systems, highlighting risk and risk management issues, or tracking changes are likely to be challenging, even with manual prompting. Finally, even when focusing on the code itself, the accuracy of the documentation provided can be highly dependent on the user. Themisto has been tested on popular datasets and machine learning modules, using common naming conventions in the field. If the user can choose different names and not follow the convention of usage, the generation of documentation may also prove to be more challenging [WWD+22].

5.4. On the Conundrum of One Model vs. Multiple models

As a final addition to the chapter, the benefits and drawbacks of using one or multiple models for the documentation is to be discussed. All the different notations above may have shown, that it is probably impossible to cover all requirements within one chosen type of media, irrespective of the type of media chosen. Even though it is a general statement, but interpreting the analysis of the introduced options, visual means generally help to meet requirements, such as transparency and comprehension, whereas textual means appear almost mandatory to capture all details, if required. Conclusively, this may suggest that combining multiple options would be logical.

However, while the accuracy of such a statement undoubtedly depends on the scenario to which it is applied, some general limitations must also be borne in mind. Firstly, different attention spans and time restrictions should also be taken into account. Again, in a plethora of scenarios, such as legal audits, having a detailed and fleshed out documentation is paramount. For many others, however, extensive documentation can lead to misunderstandings and disinterest in the product, as the sheer volume can be overwhelming. Adding to this point particularly is the potential requirement of having to learn multiple model notations. Such behaviour may lead again to errors. While showing the same concept from different perspectives can help to deepen understanding, in many cases it can be a challenge to model in a way that allows seamless switching between different models of the same concept. Sometimes, it may even lead to inconsistencies in the elaboration, which perpetuates challenges in comprehending the matter fully. Another detail to add to this list is the additional effort required to write and maintain additional documentation [AF21].

Ultimately, while the answer may not seem satisfactory, each individual use case needs to be examined and a decision made based on the underlying constraints. For projects that may be working with sensitive data and whose use may be considered risky to some extent, it may be mandatory, and in some cases legally required, to have extensive documentation on issues such as risk prevention, incident management and accountability, in addition to describing how the AI system works [KT22], [CC22], [HMD+23]. Conversely, for small research projects that are not considered risky, a diagram that provides an overview and allows others to quickly grasp the idea may be sufficient and even advantageous for gaining further support [KPL+14], [AF21].

6. BEAM: A support tool for AI System Documentation

Based on all results gathered so far, the final part of the thesis with he the task of answering RQ3 can be started. To explore different potential automation options, several scripts acting as a proof of concept have been developed. However, to also keep the previously found results and their strengths in mind, the extend of automation possibilities will be tested against an extended version of an already existing notation. This allows to set an initial step towards meeting the detected requirements better, as well as allowing to provide some potential necessities facilitating the setup for developing a toolkit to work with said notation. Thus, firstly an extension of an already established notation will be presented, which tackles a section of the discovered requirements. To this end, we use the Boxology V2 presented by Van Bekkum et al. [VHTT19]. as our base notation. The proposed extension has been coined as *Boxology's Extended Annotation for Machine Learning Systems*, or *BEAM* for short. It aims to meet some of the requirements that were previously considered to be not fully met. While in an ideal scenario as many requirements of different stakeholders as possible can be satisfied with one documentation notation, some requirements diverge to steeply tbe satisfied sufficiently. As such, while still keeping all found needs in mind, for the following pioneering attempt, the focus is majorly set to fulfill the requirements of *producers* and the *academic sector* better. This two groups represent the most direct connections to the first groups potentially using BEAM in the future, or even developing it further in the future.

6.1. An Overview of The BEAM Approach

As the Tables 4.2 and 4.3 show, both stakeholder groups share almost the same workflow- and data-requirements. In terms of ecosystem requirements, the produces are more focused on risk and risk management, and both envisage an option to display accountability and incident management. For the documentation structure, especially the academic sector promotes clear documentation guidelines and structure within the documentation. Concerning the non-functional requirements, both stakeholder focus on transparency, scalability, granularity, unambiguity, and modularity. In addition, a feasible balance between flexibility and a low cost of learning has to be considered as well.

To cover all these requirements in, BEAM comprises of multiple components. Firstly, the aforementioned extension of the notation will be covered. To do so, a brief comparison of different editing tools will be made. On top of the already existing requirements, some technical requirements relevant for the scope of this work have to be taken into consideration. Said technicalities will be elaborated in the next section 6.3.1. Additionally, to keep RQ3 in mind, as well as to further boost transparency and usability, a set of scripts have been developed. To provide an initial insight on the inter-workings between the notation and the programmed tools, a brief overview has been created and is observable in figure 6.1. The details concerning the development of the prototype components are explained in the upcoming section 6.3.4. All the content can be found in the repository of the thesis [1].

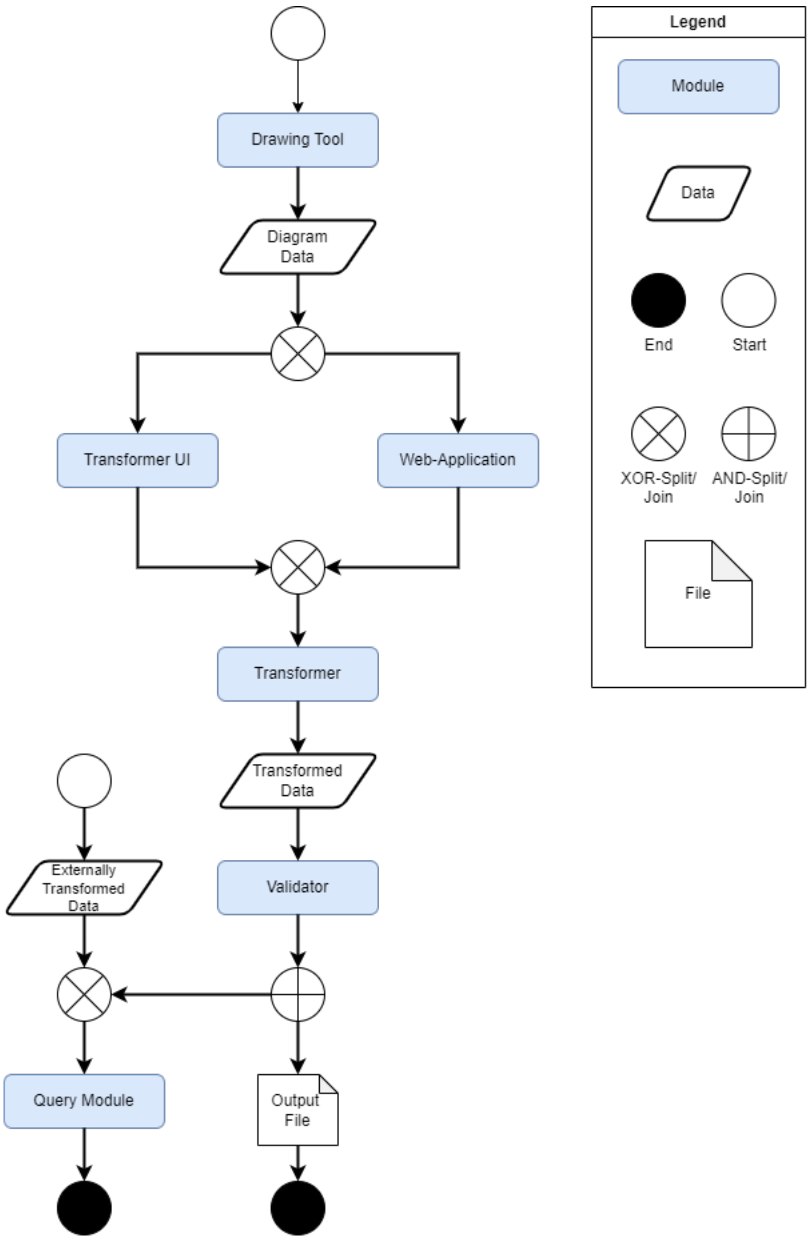


Figure 6.1.: Beam Workflow

6.2. Conceptual Elaboration of Beam

With all necessary components to understand the motivation for creating BEAM covered, the next step is to explain the overall concept of BEAM. To achieve a sufficient explanation and to allow reproducing the approach taken in this work, each of the components and their purpose are elaborated.

[1]https://git.wu.ac.at/semsys/master

6.2.1. Notation Extension

In the following chapters, the main ideas and their derivation are elaborated in an abstract way. As such, the concept description entails the notation extension, the automatic extraction of diagram data, the validation of said data, an option to query the extracted data, as well as a web-component offering the entire service.

Beam Notation: The extension of the notation serves to provide a better coverage of the requirements. As indicated in section 5.2.1 , the issues of confusion due to different levels of granularity, the addition of other elements as required, and the linking of the workflow to other concepts that may be required need to be addressed. According to the requirements, this is best achieved by providing a flexible, transparent, easy to learn and explain extension to the notation that is applicable at different levels of granularity.

In practices, these findings have been translated to multiple extension ideas. The first one would simply be to provide the option of introducing new elements. This measure would provide the flexibility required. That being said, it may reinforce the potential of confusing the reader. To avoid this situation, measures should be introduced to explain the elements used and to allow switching between different levels of granularity. Another challenge to be addressed is the low cost of learning. Again, this can be addressed to some extent by explaining each used element in the system. However, considering that AI systems may require numerous elements, the representation of such a system can quickly become convoluted. Therefore, the actual number of newly introduced symbols should also be minimised.

To cover the aforementioned needs, BEAM introduces some new elements which extend the base template of van Bekkum and colleagues [VHTT19]. These element entail basic options to attach arbitrary text to each element, as well as extending and collapsing elements that are tied to a ”main” element. To clearly distinguish between the base workflow, some additional connector options have been established as well. Furthermore, BEAM introduces the requirement to add a legend explaining each element by adding the shape and tag of the element into said legend.

6.2.2. Beam Components

Data Extraction: To answer the question of the degree of automation applicable for documenting AI systems, the potential options are to be illustrated. The idea for BEAM in this case is to take the data from the diagram that is stored in the background and process it into other ”easier to process” formats. To accomplish this step, an editor that stores the data not only in an image format, but in any processable data structure is highly advantageous. Technically, it would be possible to find the required information via image processing, but this is not particularly practical and is beyond the scope of this work [CWG+21]. Once the data can be accessed, the consecutive process would be the extraction of the data in the format desired. Depending on the actual data structure, this step may require different algorithms. In terms of storage, the information is stored in a simple output basically appending each found node as new point into an output file.

Validation: Another step, although not strictly necessary, are some simple data validation algorithms. These validations are mainly introduced to perform some basic syntactic and semantic checks. They currently include to check if each element has its tag assigned, the original elements are still understood as such based on a static configuration file in the background, and that at least one system has been modelled in the information provided. In addition, another small check has been introduced, which tries as best it can to see whether the model is consistent with the main notation [VHTT19].

Querying the Data: In terms of querying the data, a separate module has been opted for the best solution. This solution allows you to process one or more files that have already been through BEAM's data extraction process. Based on the input, the data can be queried using either custom queries or some basic options. The basic options cover queries such as finding certain elements that share a characteristic, or finding the longest workflow within the process.

Web-Application: The final component offers a web service of the aforementioned modules for convenience of usage. The web service allows to upload a file and return the result of all steps mentioned before.

6.3. Prototype Implementation of BEAM

With the concept of beam elaborated, the overall architecture of Beam can be illustrated. Subsequently, the technical details about the notation, and the scripts developed are going to be introduced. Beam is designed to be adaptable and modular for different scenarios of usage. More descriptively stated, the notation itself is decoupled from the validation code of the prototype, as well as any editor. The additional elements are designed to be "generic" An example can be seen in figure 6.2. The overall objective is to allow usage of the notation in other settings outside of this work, if so preferred.

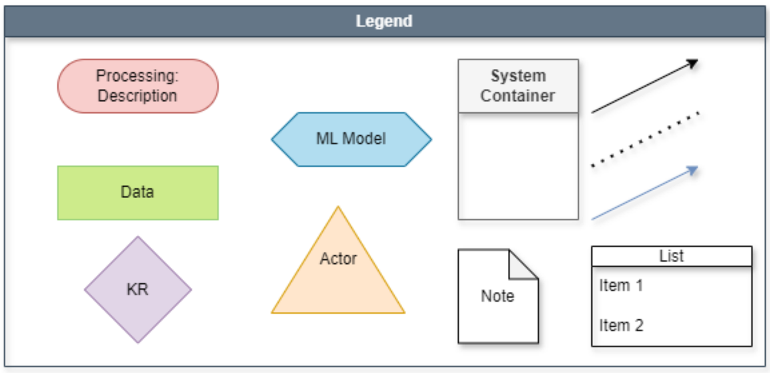


Figure 6.2.: Example of the Beam Notation Elements

Moving on to the developed scripts, from the he technical perspective, the scripts follow the technical explanation of the notation extension. This is mainly to show to what extent the BEAM notation can be automated. As for the process itself, automation is achieved by extracting the content of the diagram(s) provided, restructuring it from a tree of elements to a "simpler" data-structure, and by checking for errors. Additionally, a simple web-application providing these services in form of a website has been introduced as a concept to offer easy access to the toolkit. However, to get to the point of automation, one component not mentioned until now is missing. To create a diagram, a suiting editor is required. As such, finding the righr editor for the scope of this work is covered in the next section.

6.3.1. Comparison of Diagramming Tools

With the notation covered, a brief overview of existing options to create and edit different diagrams is provided as well. In addition to the requirements found and the requirements the notations introduce, there are some technical details to consider as well. Specifically, one of the content produced for this thesis includes Python scripts [VRDJ95]. As such, on top of all other requirements, the software used has to to have certain features included. These additional features are specifically required for the scope of this particular work. Of utmost importance is the ability to import and export diagrams in a form of machine-readable-format (JSON, or XML for example). Furthermore, additional requirements entail the solution being freely available to allow all users to try the solution of this work (ideally open source), and the ability to include custom elements, should they be required. Other non-mandatory requirements, which would be ideal to have regardless, are the option to create and use templates, synchronous collaboration, cross-platform-support (or a web-application), integration/embedding options into other apps, such as Confluence [Atl24], and/or an API [RVS+23], [HMV+22]. For the upcoming comparison, only the technical requirements will be considered.

6.3.1.1. Draw.io

Draw.io is freely available, fully released, and open source general purpose modeling tool offering the capability to create various types of diagrams. The tool can either be used by installing a local version, or by using the website of Draw.Io citedrawio. For the local version, installer for commonly used operating systems are provided [DD24].

To create models, a drag and drop system of elements has been implemented. A plethora of atomic elements, such as simple rectangles, up to complete templates for quickly displaying entire processes are offered in the base version. In addition, the option to load pre-defined templates, as well as external elements (assuming they adhere to the rules of a MxGraph) are offered. In terms of machine-readable format, the core structure follows a MxGraph and is typically stored in an XML file. However, other options, such as storing the diagram as a readable URL are provided as well. Said format can be read as well as exported and thus offer an option to interact with the tool [Dra23].

That being said, whilst at least for the local version a CLI has been developed, there is no native support for an API. Draw.io allows the embedding in other services, however. Thus, if an objective of the user is to fully automate the entire process from model plotting to converting into a machine-readable format, the according commands would have to be triggered externally. Moreover, team collaboration can only be conducted in an asynchronous fashion [Dra23].

6.3.1.2. Apache Open Office Draw

Similar to Draw.io, Apache Open Office Draw (*Office Draw* is used as a short form from this point onwards) is a free to use tool with its core being open source. It may also be considered a general purpose modeling tool not focusing on a particular type of diagram. However, to the time of research, no readily available web-service has been found. Moreover, Office Draw is a component of the *Apache Open Office package*. Thus, it has to be downloaded and may result in users having additional unwanted software installed [Apa23].

In terms of base functionality, Office Draw is comparable to Draw.io yet again. More elaborately explained, it features a drag-and-drop UI with adjustable parameters per element and allows for swift creation of diagrams. As a general modeling tool, numerous different elements can be chosen as symbols to represent the required components of a machine learning system. Furthermore, the import and export of templates is implemented as feature [Apa23].

Moving to the additional requirements apart from the modeling aspect, Office Draw allows for exporting XML and thus provides an option for further processing a created model. Concerning support on different devices, the Apache Open Office package supports commonly used operating systems. However, as distinct software tailored towards the framework of Apache Open Office, it likely can not be easily embedded into other software or tools. Finally, whilst an option to share the same document among different users is provided, only one user can manipulate one file at the time. Consequently, collaboration is only possible in an asynchronous fashion.[Apa23].

6.3.1.3. Mural

Mural is primarily web-application focused on synchronous collaboration in teams. That being said, local downloadable versions for Windows and IOs are also available. No support for Linux is not mentioned, however. Contrary to the other options presented so far, there are multiple tiers to consider. Per se, it is a commercial software, although a free version for personal use is available [2]. However, like Draw.io or Apache Open Office, Mural is not focused on a specific type of models and offers a sizable range of elements to draw with [Mur23]. Moreover, previous works have either mentioned Mural or used it for their projects [HHM+20] [PPW+21].

In terms of technical capabilities, some differences to the other software are observable. Starting with the templates, per se built in templates are offered. However, it is unclear, whether custom templates can be created and shared for usage by other members. Furthermore, it is also not determinable if custom shapes can be created. Regarding portability, there also appears to be no option to import and export diagrams in a machine-readable format. However, Mural offers an API for communicating with the application, which leads to the inference that data of diagrams is transferable. It also has to be annotated, that the API is tiered and only the highest tier opens the API in its full functionality. Finally. Mural may also be integrated into other applications, if desired so [Mur23].

[2] `https://www.mural.co/pricing`

6.3.1.4. Creately

Creately is another option found during direct search for diagram editor tools. Like Mural, the company behind Creately offers a free and paid version. The free account allows to create up to three ”canvases” with a maximum of 60 elements simultaneously per canvas [3]. Furthermore, it appears to be an online tool only and also focuses on peer collaboration [Cre24].

Moving onto the technical details, a basic support of elements is provided. Plotting is done via a simple and intuitive user interface. Concerning the import and export capabilities, little information was found. It is also unclear, whether an available API exists. For the ability to create and share templates, a selection of pre-defined templates is offered. However, it appears that it is not create or work with custom templates in any form. Finally, in terms of integration with other tools, no indication could be found of any option for integration with other tools. [Cre24].

6.3.1.5. Knime

The final candidate found would be Knime. One feature that sets Knime apart from all the other tools found is its focus. Rather than being a general purpose editing tool, Knime focuses on workflows and setting up workflows and ETL processes. Thus, for subject matter experts, Knime may be the apparent option. However, for the scope of this work, the cost of learning the tools also has to be considered. Per se, Kime is open-source tool with all core functionalities available to the user. However, Knime offers also a tiered subscription, where the selected tier offers an extending list of features, such as API access. Knime supports Windows and CentOs as operating systems. [Kni23].

In terms of aptness for this work, the first ”issue” paradoxically would be Knime being a workflow tool. As such, setting up workflows is done in a controlled setting with many pre-defined elements. While Knime may be used to model diagrams, it is not the main purpose and may have to adhere to additional restrictions. Nevertheless, Knime also supports plugins and custom extensions, which may be used to help creating diagrams. Furthermore, Knime also supports template support and even a website to publish templates [4]. In terms of import- and export-capabilities, Knime offers these options, though only in their own file format. Collaboration can also only be conducted in an asynchronous fashion. Finally, Knime can also be embedded into other application as well [Kni23].

Overview of the Comparison

As final section of the comparison of the tools, the key parameters and the tools are displayed in Table 6.1. All of the features presented in the introduction have been grouped into three priority levels, with the first level considered essential, the second important and the third additional features that may help this work. One annotation is to be made. The perceived ”easiness of use” and ”intuitiveness” are subjective scores and may be discarded for objective reviews.

[3]https://creately.com/plans/

[4]https://forum.knime.com/t/knime-os-compatibility/12677

Tool	Criteria of Evaluation										
	Priority 1			Priority 2					Priority 3		
	Freely available	Import/Export capabilities	Customisation of elements	Easiness of use	Intuitiveness	Option to save/load templates	Synchronous collaboration	Cross platform support	Integration/Embedding options	(Freely usable) API	
Draw.io	✓	✓	✓	2	1	✓	×	✓	✓	×	-
Open Office Draw	✓	✓	✓	3	3	✓	✓	✓	✓	✓	-
Mural	∼	✓	✓	2	1	∼	✓	∼	✓	∼	-
Creately	∼	∼	✓	1	1	∼	✓	✓	∼	∼	-
Knime	∼	✓*	∼	4	5	✓	×	∼	✓	✓	-

”-”: No information found; ”∼”: Only partially featured/available

* In a custom file-format

Table 6.1.: Comparison of Tools for Creating Diagrams

6.3.2. Summary of the Design Decisions for the Implementation

Based on the comparison in Table 6.1 and all other detected requirements, the setup for the proof of concept is to be determined. With all requirements covered, the chosen setup and the rationale for said decision can be elaborated.

Starting with the chosen notation, as already briefly covered in the introduction of this chapter, a boxology has been selected as modelling method. The main reason for this decision entail the already defined focus on AI systems, the simple design, as well as the option to plot at different abstraction levels already being part of the base notation [VHTT19]. As pointed out by the authors of the boxology, the model elements are specifically adapted for AI- and hybrid-systems. This fact allows to model the workflow of AI systems more precisely. Furthermore, by considering hybrid systems, the inclusion of external elements can be facilitated, if necessary. Moving on, the option of modeling the systems as abstract base class, as well as an actual instance thereof, at least two levels of granularity are covered. Finally, the use of basic shapes allows modelling in different editors and is likely help to render the systems more transparent, especially if the AI system itself may include numerous components which are to be modelled.

As for the editor, table 6.1 displays two viable options for the scope of this work, which would be *Draw.io* and *Open Office Draw*. There are distinguishing advantages for both alternatives. However, in order to have the option for testing the content produced, Draw.io may be arguably the more convenient option, as it is available as a local installable version, as well as an online editor. The online version offers full functionality and thus no software has to be installed in most instances, if not wanted [Dra23]. Even though the arguments may be considered subjective to some extend, they have lead to the decision to develop the upcoming notation and scripts based around Draw.io.

6.3.3. Notation Extension

With the reasoning behind the upcoming implementation covered, the first logical item to start with would be notation itself. The chosen boxology, as well as the proposed extension, are kept in a minimalist design to allow for easy replication. All elements can be found in Table 6.2. Each of the added elements and methods originally stems from other notations. The notes and lists originate from various UML diagrams [MLNN20], [HG22], [JPZ22], [AF21]. As for the idea of including different types of flows and connectors, this idea is derived from the message flow of BPMN [CT12]. For each workflow element, a set of default properties have been defined as well. One example can be found in Figure 6.4. Additionally, to reduce the additional cost of learning for users, the syntactic rules have been compressed as well. However, the option of flexibility still are to be met to a sufficient degree. This conundrum may be partially soluble with an "opt-in-system". In practice, this opt-in-system envisions using a legend for defining the syntax of the current diagram. In more detailed terms, one of the added rules would be to add a legend to each diagram created using BEAM notation. In the legend, each element used is included by adding the "form" to the legend and labelling it according to its purpose. If necessary, even existing standard elements of the existing elements could be repurposed to fulfil a different role.

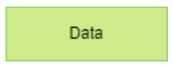
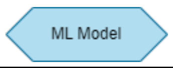
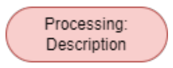
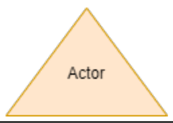
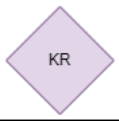
This is intended to make the diagram easier to understand, even for reviewers who are seeing it for the first time and are unfamiliar with boxology or BEAM notation. However, once again, flexibility comes at a price. The ability to add own elements and use existing elements in a different way, reviewing multiple models might be confusing and a potential source of error. Reviewers may not check the legend in detail and assume that each element has the same task seen in previous diagrams. However, to test if additional flexibility in form of alteration of the original elements is required in some scenarios, the option to alter said "default elements" has not been ruled out. Nevertheless, even for this early exploratory stage, it was still recommended to keep the provided default elements as they are.

Going back to the general scope of the diagram, if the other components developed for the scope of this project are to be utilized, some additional rules are to be followed. To further elaborate on these rules however, another mechanic has to be introduced first. As one of the key requirements is granularity, a system should be able to display different information at different levels of detail. To cater for this requirement, BEAM introduces different workflows. The first workflow marked by a black arrow indicates the general workflow of the modelled AI system. The process usually starts with data to import and walks through the system until the desired output is achieved. However, in order to take into account the granularity and the fact that there are several requirements that revolve around the ability to attach different types of information at different levels of detail, two additional auxiliary connectors are introduced. The first one takes the form of a dotted connector arrow. The purpose of this connector is to allow leaving comments to either individual elements, multiple elements, or systems.

The second connector has been named "*Information Attachment Connector*". Its purpose is similar to the comment connector. However, contrary to a connection to comment, this connector is designed to have arbitrary elements, group of elements, or even entire subsystems connected to it. This content can be used to either elaborate individual items in more detail (e.g. to describe the structure of on ML-Model), or to attach other relevant information (e.g. a system explaining the process of risk prevention and mitigation for the input data could be attached). When using the exact setup chosen for this project, there also is the option to collapse and expand the content via on press of a button.

Returning to the additional rules, to facilitate this behaviour in the validation, some additional requirements are necessary to work with the prototype. Firstly, each element in the diagram has to set a property indicating which element it is. This measure allows elements to be labelled arbitrarily. Additionally, it provides a more robust approach for querying and validating the content with the scripts developed.

The instructions to use the notation and the prototype are covered in the linked repository [5]. Summarized, the easiest way to use BEAM would be to load the provided *beam_lib.xml* library into Draw.io and use the elements created. Within the library, all elements of the core boxology, as well as the proposed extensions can be seen. An overview of the elements is also observable in Figure 6.2. Should the provided selection not be sufficient, custom elements may also be added. As long as the element can be found in the legend, and the new elements have a "element" property indicating how the element can be found in the XML-File (e.g. "element: ml_neural_network" to designate a specific elements for neural networks solely), additions to the library should work.

Designation	**Description**	**Element**
Data	Displays any form of data or other information in the workflow	
ML Model	Any type of machine learning model	
Process	Process or "action" performed between two other non-process elements	
Actor	A role occupied and responsible for one or more tasks in the system	
Knowledge Resource	A form of external knowledge storage introduced to the AI system	

[5]https://git.wu.ac.at/semsys/master

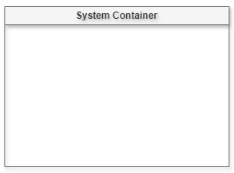
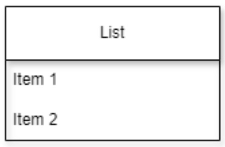
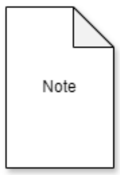
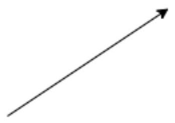
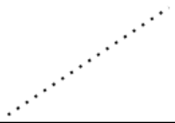
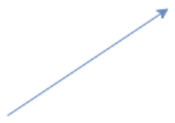
System	Container housing the AI system or a component thereof	
List	List of characteristics or other information associated with the element to which it is attached	
Note	Arbitrary content in textual form attached to any other element	
Workflow Connector	Shows the workflow/data flow of each step in the AI system	
Dotted Connector	Connector for connecting notes with other elements	
Information Attachment Connector	Indicates that further details (usually at a different level of granularity) are attached to this element. Also indicates that this information should be hidden, if necessary, in order to better display the main workflow	

Table 6.2.: Elements of Beam

Finally, to provide an opportunity to see the notation in use, the example shown in Figure 6.3 can be observed. Said example displays a simple AI system where two types of neural networks are used to predict credit scores. Furthermore, it displays the legend with all possible default elements, as well as attached details.

In addition, to highlight how default properties are attached in the BEAM notation, an example of the default properties of an ML Model is presented in Figure 6.4. Furthermore, this example presents how a selection of default characteristics for one specific property can be offered. This notation is inspired by UML Class Diagrams [Rum16].

6.3.4. BEAM Components

Moving on to the scripts developed for BEAM, a short functional description of each component and its purpose will be provided. For a detailed elaboration, please refer

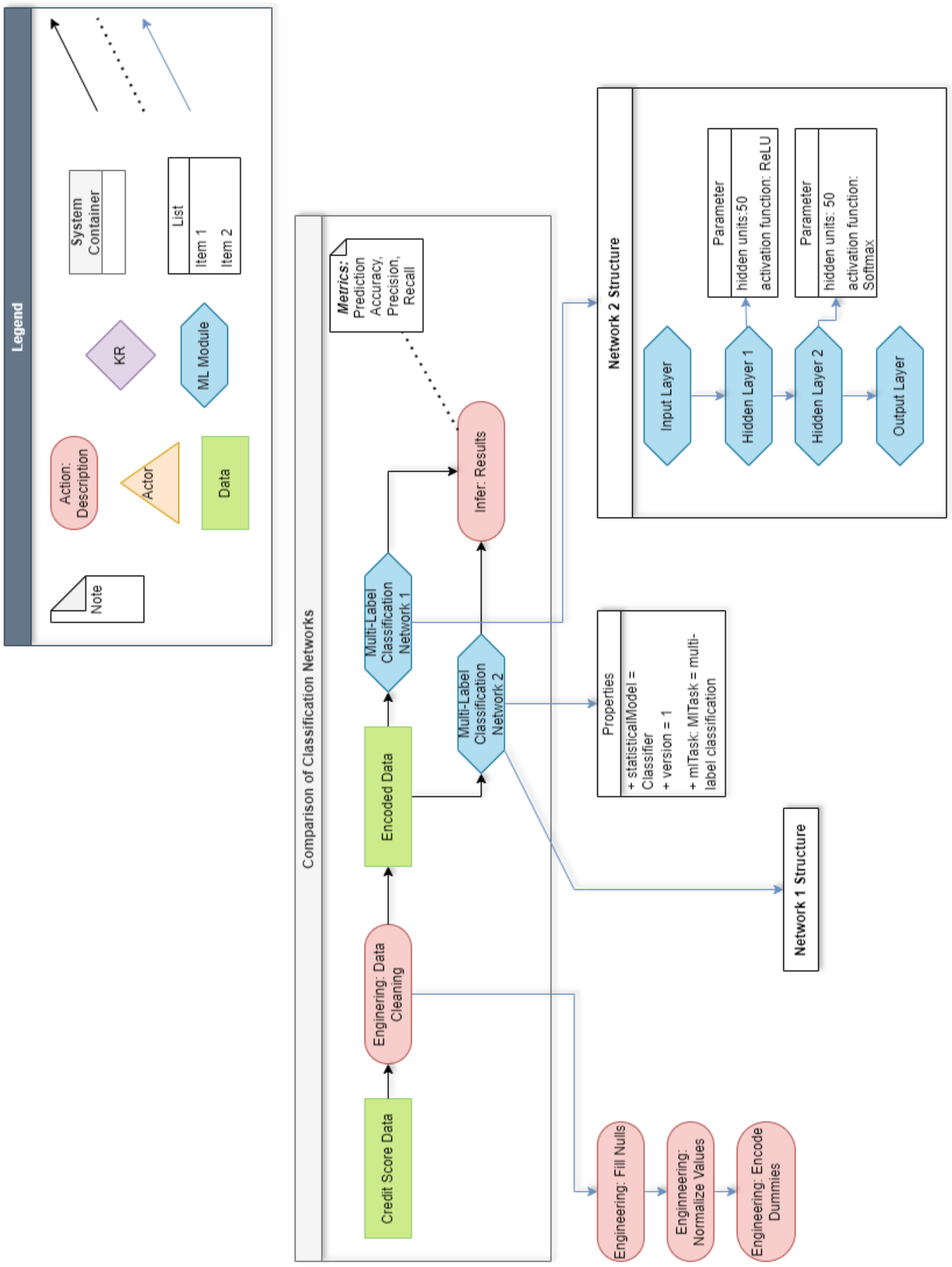

Legend
Note
Action: Description
Actor
Data
KR
ML Module
System Container
List
Item 1
Item 2
Comparison of Classification Networks
Credit Score Data
Engineering: Data Cleaning
Encoded Data
Multi-Label Classification Network 1
Multi-Label Classification Network 2
Infer: Results
Metrics: Prediction Accuracy, Precision, Recall
Network 2 Structure
Input Layer
Hidden Layer 1
Hidden Layer 2
Output Layer
Parameter
hidden units:50
activation function: ReLU
Parameter
hidden units: 50
activation function: Softmax
Properties
+ statisticalModel = Classifier
+ version = 1
+ mlTask: MlTask = multi-label classification
Network 1 Structure
Engineering: Fill Nulls
Enginneering: Normalize Values
Engineering: Encode Dummies

Figure 6.3.: Example of an AI System Modelled in Beam

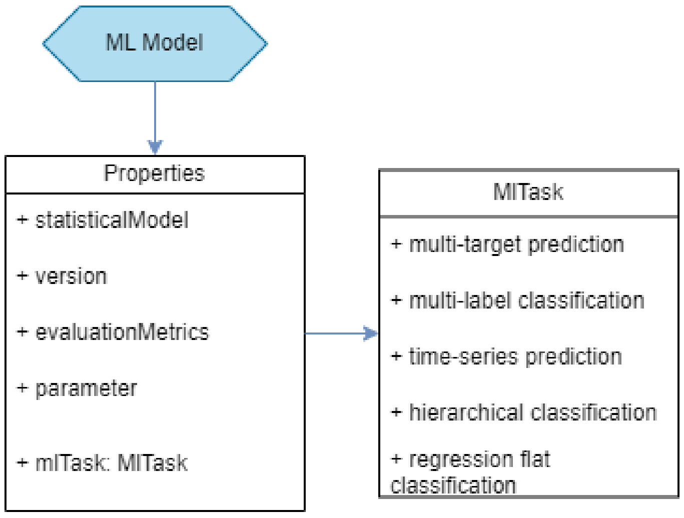


Figure 6.4.: Example of the Default Properties of an Machine Learning Model Element

to the accompanying repository [6]. As already briefly indicated, the whole concept of BEAM is set up in different mostly independent modules. As such, to work with the code, there are two options. The first option would be to download the repository and set up an execution environment for Python (i.e. a Python interpreter with all dependencies installed) [VRDJ95]. Alternatively, a small Flask-Application has been set up . Said application allows to import a file, conduct the necessary transformations, and provide the output again. However, the Flask-App would have to be hosted by a machine [Gri18]. The remainder of this section walks through the logical steps of the code as it would be executed. A brief overview of all the Python modules can be seen in Figure 6.5 [VRDJ95].

For the main function itself, the two different modes can be set. The first mode *"extract"* can be used to extract the content from a file. Conversely, the second mode *"query"* can be used to query one or more previously extracted files.

Configuration: The first step of each new instance is to apply the configurations. Within said configurations, almost each aspect of the program can be controlled. One example displaying the default settings of the validation component can be seen in the Listing 6.1. The configuration file hereby acts as the file containing the default configurations used during development. For each instance, there are two option to overwrite the defaults. The first one would be to provide parameters upon calling the main function. The second option would be to call the main function via the accompanying CLI-Tool and providing the arguments there.

[6]`https://git.wu.ac.at/semsys/master`

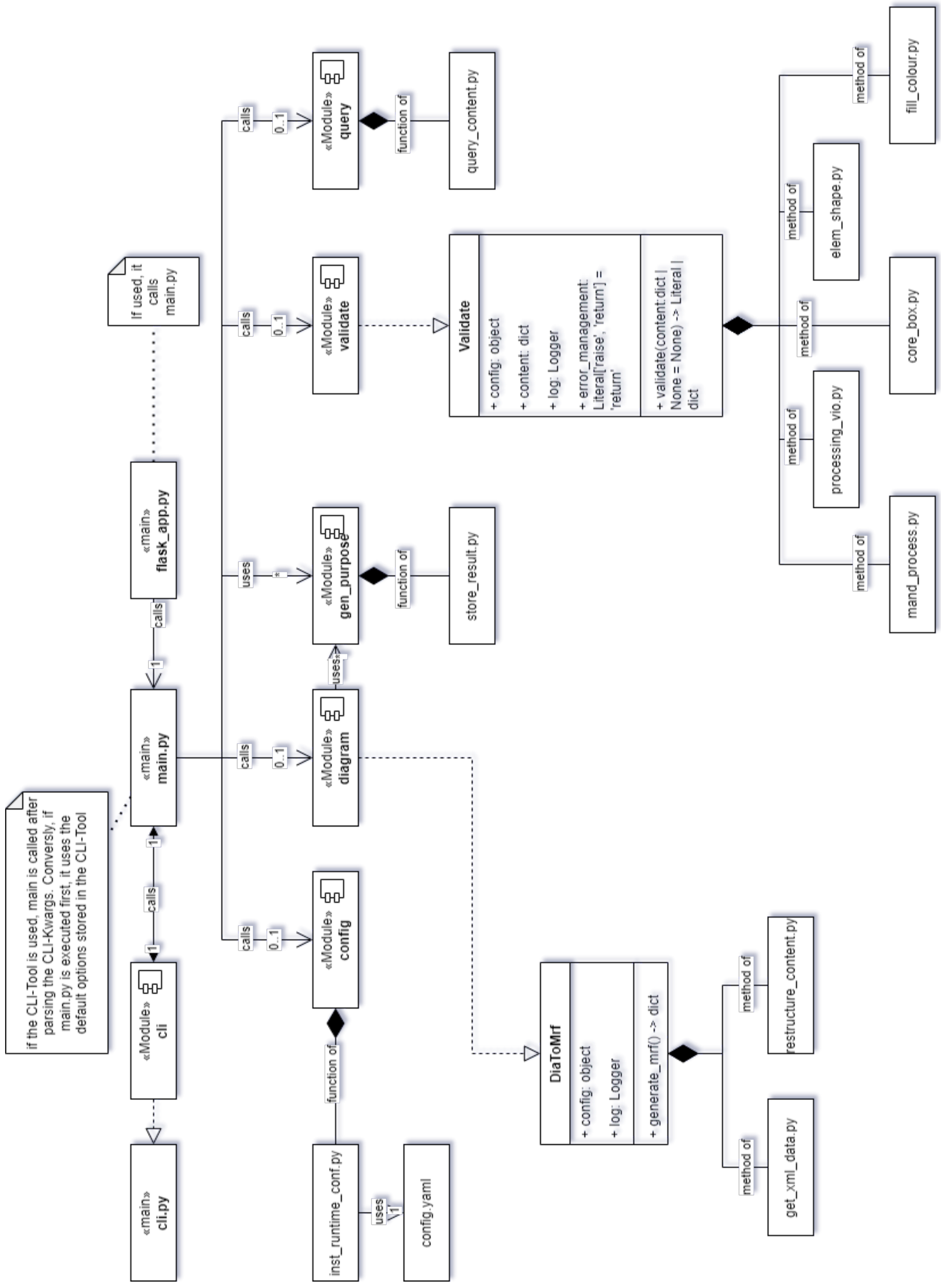


Figure 6.5.: Architecture of the Python Modules

```
validate:
  # Defines whether the usage of elements considered
  # core boxology shall be enforced
  enforce_strict_core_elem: False

  # All checks to conduct
  checks:
    strict_core_boxology: True
    mandatory_elem_prop: True
    process_violations: True
    mandatory_elem_fill_colour: False
    mandatory_elem_shape: False
```

Listing 6.1: Section of the adjustable options in the configuration file "config.yaml"

Convert to a Machine-Readable-Format: One of the core functions of the scripts is to take a supplied XML file containing a diagram created in Draw.io, extract the relevant content and restructure it for easier processing. XML files are arguably already structured in a well-established data structure, namely a hierarchical tree. However, given the nature of the diagrams drawn, the content resembles more a graph, rather than a hierarchical structure. Thus, it is easier to extract each element, store it in a simple dictionary, and follow the data-flow of the systems. Exactly this process is the purpose of this component.

In practice, for the scope of this work the result is twofold. Firstly, the extracted content is stored in an intermediate file to check if all the content has been extracted correctly. An example of a section of this step can be seen in Listing 6.2 Secondly, the content of said intermediate file is used to walk through the provided tree and restructure the content. The default output developed for this content works by first finding all systems marked as system (including the legend), and attaching each element the found systems. Each element not belonging to a system is attached as independent element. Once done, the two files containing the extracted tree, as well as the restructured content are returned.

```
<root>
   <mxCell id="0" />
   <mxCell id="1" parent="0" />
   <object label="data: Procedure&lt;div&gt;
        Data&lt;/div&gt;" element="data"
        sensitive="False" id="lizAr8-d5KSNGjCRc8-C-24">
     <mxCell style="rounded=0;whiteSpace=wrap;html=1;
            fillColor=#cdeb8b;strokeColor=#97D077;
            " parent="1" vertex="1">
       <mxGeometry x="940" y="940" width="120"
                height="40" as="geometry" />
     </mxCell>
```

```
    </object>
    <mxCell id="lizAr8-d5KSNGjCRc8-C-28"
    style="edgeStyle=orthogonalEdgeStyle;
    rounded=0;
        orthogonalLoop=1;jettySize=auto;
        html=1;entryX=0;entryY=0.5;
        entryDx=0;entryDy=0;" parent="1"
        source="lizAr8-d5KSNGjCRc8-C-17"
        target="lizAr8-d5KSNGjCRc8-C-27" edge="1">
      <mxGeometry relative="1" as="geometry" />
    </mxCell>
    ...
```

Listing 6.2: Section of the exported XML file of an AI system modeled in BEAM

After restructuring, the output would look as depicted in Listing 6.3.

```
{
    ...
    "object_1": {
        "label": "data: Procedure<div>Data</div>",
        "element": "data",
        "sensitive": "False",
        "id": "lizAr8-d5KSNGjCRc8-C-24",
        "x": "940",
        "y": "940",
        "width": "120",
        "height": "40",
        "as": "geometry",
        "style": "rounded=0;whiteSpace=wrap;html=1;
                    fillColor=#cdeb8b;strokeColor=#97D077;",
        "parent": "1",
        "vertex": "1"
    },
    "mxCell_3": {
        "id": "lizAr8-d5KSNGjCRc8-C-28",
        "style": "edgeStyle=orthogonalEdgeStyle;
        rounded=0;orthogonalLoop=1;jettySize=auto;html=1;
        entryX=0;entryY=0.5;entryDx=0;entryDy=0;",
        "parent": "1",
        "source": "lizAr8-d5KSNGjCRc8-C-17",
        "target": "lizAr8-d5KSNGjCRc8-C-27",
        "edge": "1"
    }, ...
```

Listing 6.3: Section of a restructured AI system

Validate: Once the extraction process is complete, the data proceeds to be validated. The validation content is versatile in nature, as most of the processes are optional. The one stable routine in each instance is the extraction of the legend and checking whether each element is present in the legend. Otherwise, every other check can be activated, or deactivated as seen fit. For each check, if an error is found, said error will be attached to a temporary storage. Once all checks are finished, the data-structure containing all collected errors will be returned. Should there be no error, a simple string stating ”no errors found” will be returned.

Per default, the check which enforces the core boxology, whether each element has the property defining its purpose, as well as checking for violations in the process are active. Starting with the check enforcing the core boxology, it takes the statically stored properties that characterise the core elements of the original boxology, and compares each element having its property set to one of these core elements. If the defined characteristics do not match (for example, the shape or colour of the element), an error is added to the error list.

Advancing to the property check, as the name may indicate, this check takes every element and checks whether the element-defining (*element: ...*) property has been set. Should any element not having this particular property assigned occur, it will be included in the error storage.

Finally, the process violations check checks for syntactic and semantic errors in the usage of processing elements throughout the diagrams. The premise of this work has been to adhere to the core boxology and effortlessly weave in the extensions. As a result, the validation essentially checks for adherence to the core boxology. There are still functions that are required to distinguish between different elements, such as distinguishing between different connectors. However, for the time being, no additional validation has been developed for all the extensions. An example excerpt of found errors can be found in Listing 6.4.

```
{
    # Missing the "element" property
    "Mandatory element missing 0": " The mandatory item \"element\"
    is not present in: {'id': 'lizAr8-d5KSNGjCRc8-C-24'} ",

    # Missing the assigned shape to the element
    #(derived from the legend)
    "Mandatory element missing 1": " The mandatory item
    \"['shape', 'rounded']\"
    is not present in: {'id': 'lizAr8-d5KSNGjCRc8-C-25'} ",
}
```

Listing 6.4: Example of found errors

Flask-Application: In order to allow users to quickly use the scripts provided, a small web application has been written to show how the service can be offered via a public domain once fully developed. To do so, the Flask-Framework has been used [Gri18]. The small application allows to upload a XML-File created in Draw.io. Once a file has been uploaded, a zip-archive containing the extracted nodes (if stored), the restructured content, as well as a file containing all the found errors from the validation module is returned. Regarding the structure of the app, only Flask and accompanying HTML templates have been used. Whilst the app is not in a production-ready state, it can be quickly tested by executing the *flask_app.py* file.

Querying the Data: Finally, the scripts also include another execution mode, namely the *"query"* option. This feature allows to query one or multiple files, which are restructured via the script restructuring the content presented before. For the scope of this particular work, querying has been limited to simple terms, such as finding a specific element via the element property, or getting all elements having a particular label. The query parameter can be relayed by invoking a command line tool.

6.4. Evaluation of BEAM

As a final step in the development process, initial feedback on the BEAM notation was gathered through a survey. For this section, multiple subsections are covered. First, the general structure of the survey is briefly described. Next, the survey results will be presented. The ultimate subsection covers the interpretation of these results.

6.4.1. Setup

Firstly, the setup and structure of the questionnaire is to be described. In order to get feedback from different potential users, different target groups have been asked to test the BEAM notation and provide feedback. These groups include students, instructors, professors and people in other roles such as professionals. Concerning the structure, the questionnaire is adapted from the TAM model [AEMK18]. Within the survey, three sections regarding the content are covered overall. The first section revolves around the perceived usefulness, while the second section focuses on ease of use of BEAM. Finally, the final section consists of two open-ended questions about experiences with the notation and feedback on aspects that could be improved. In addition, a small section on demographics was added at the beginning.

6.4.2. Evaluation Results

In the next section, the answers to the questionnaire used for the evaluation of BEAM can be found. It includes the open-ended questions, the numerical responses to the TAM sections of the questionnaire and the analysis of these numbers, as well as general demographics. The numeric results are depicted in Table 6.3. All answers to the open questions are following the numeric results. As for the demographics, the result is shown in Table 6.4. All questions used in the questionnaire can be found in the appendix in section B.

Interviewee /Metric	**Perceived Usefulness**									**Perceived Easy of Use**						
	[A001] Efficiency & Communication	[A002] Documentation Quality	[A003] Productivity	[A004] Documentation Comprehension	[A005] Ease of Use	[A006] Usefulness	[A007] Team Coordination	[A008] Communication Stakeholders	Overall	[A101] BEAM vs. Text Description	[A102] Ease of Use	[A103] Intuitiveness	[A104] Flexibility	[A105] Custom Extensions	[A106] Handling	Overall
P1*																
P2	3	3	5	2	2	4	3	2		5	4	3	4	2	2	
P3	3	3	5	3	3	3	3	3		5	4	3	3	2	3	
P4	3	2	3	2	2	3	2	3		2	3	3	3	3	2	
P5	3	3	3	3	4	3	3	2		2	4	3	5	2	3	
P6	3	2	3	3	2	1	2	2		2	3	2	3	3	2	
Average	3	3	4	3	3	3	3	2	**3**	3	4	3	4	2	2	**3**
Median	3	3	3	3	2	3	3	2	**3**	2	4	3	3	2	2	**3**
Minimum	3	2	3	2	1	2	2	1	**2**	3	2	3	2	2	**2**	
Maximum	3	3	5	3	4	4	3	3	**0**	5	4	3	5	3	3	**5**
Skew	0	-0,61	0,61	-0,61	1,26	-1,296	0,61	0,61	**0,26**	0,61	-0,61	-2,24	1,26	0,61	0,61	**-1,42**
Third quartile	3	3	5	3	3	3	3	3	**3**	5	4	3	4	3	3	**4**
First quartile	3	2	3	2	2	3	2	2	**2**	2	3	3	3	2	2	**2**
IQR	0	1	2	1	1	0	1	1	**1**	3	1	0	1	1	1	**2**

Range: 1 = Extremely likely, 2 = Quite likely 3 = Likely, 4 = Neither, 5 = Unlikely, 6 = Quite unlikely, 7 = Extremely unlikely

* The numerical values were not recorded, but the open question and demographic sections were answered.

Table 6.3.: Numeric Data - Questionnaire

Open Questions

[A201] Modelling Experience [7]:

P1 : *"While the usage and adaptation of the elements is highly intuitive and effortless, the usage of the different connection types is a bit cumbersome, especially the 'workflow connector' which is used quite heavily. When usually using draw.io, I simply connect two elements directly in the drawing board with the suggested arrows, but in this application, I need to select an arrow from the left panel every time, though this might be due to a usage error. If technically possible, it would be nice if the 'workflow connector' (as a main element) would be the standard connector."*

P1 : *"I find the option of tree folding to show/hide details extremely convenient, especially in bigger systems."*

P1 : *"There were some steps I was hesitant to model, which all ended up as Processing box, i.e., the crawling of data, performing entity linking using a KR, and the calculation of Cosine Similarity. I have the impression I misused the Processing box for at least some of these cases, as their underlying nature feels quite different. A bit more guidance would have been helpful."*

P1 : *"It would be helpful to have a nice way to directly link the corresponding resource to the elements, e.g., i.e., adding doi's to the model or data elements, or the url of the training script to the "ML Training" box."*

P1 : *"The properties, although a bit hidden, are a helpful addition, especially for more advanced users."*

P2 : *"It was nice to have the core elements already predefined so that less formatting was needed."*

P5 : *"now i dont need to define a legend each time i created a new diagram"*

P6 : *"With the fixed and standardized notation, it is much faster to draw."*

[A202] Addition or changes:

P1 *"The documentation seems a bit inconsistent, e.g., at some places Actors and KR modules are mentioned, but they are not present in the library itself. Also, the technical documentation e.g., suggests the usage of an ML component, but in the examples, Hexagons are used for all ML-related modelling. I understand that this is probably due to allowing flexibility, however, it adds a layer of confusion as the purpose of this tool is not entirely clear: is the main goal to being able to create my own modelling framework or is this a readily made framework with guidelines on how to use."*

P1 : *"One aspect that might be useful in practice is the possibility to apply a similar mechanism to show/hide the "branch" of the training procedure of ML models, as it is informative in some situations, but might overloading in others."*

[7]Due to the varying size of some responses, some of these answers have been split into several sections.

P2 : *"It would be useful to have a short description of each element and its expected usage within the editor."*

P5 : *"improve the tutorial pdf, there are couple of elements that are not explained on the pdf"*

P6 : *"More granular type of data. (for now, I represent the main input and external knowledge in the same notation)"*

Demographics

In the Table 6.4, some general information about the modelling experience, as well as the current general role of occupation can be found.

Interviewee	Modelling Experience	Experience ML	Role: Student	Role: Instructor	Role: Professor	Role: Other
P1	2	2	1	1	1	2
P2	2	2	2	1	2	1
P3	2	3	1	2	1	1
P4	2	3	2	1	1	1
P5	2	2	1	1	1	2
P6	2	3	2	1	1	1

Options: Modelling Experience -> 1 = None, 2 = Some experience, 3 = Considerable experience | Experience Machine Learning -> 1 = Foundational knowledge, 2 = Worked on some introductory projects involving AI, 3 = Worked with multiple AI systems, 4 = Considerable experience | Roles -> 1 = Not occupying role, 2 = Occupying role

Table 6.4.: Demographics - Questionnaire

6.4.3. Evaluation Results Analysis

The remainder of this chapter interprets the results of these sections. First, an abstract summary of the demographic data is provided. This is followed by an analysis of the quantitative results. Finally, the open questions are interpreted.

Initialising with the demographics, two questions regarding profession and modelling experience were asked. The survey results show that the interviewees comprise of 50 % students, some instructors and external professions, and professors. Moreover, all respondents stated that they have some model experience. Furthermore, all participants have worked on initial projects with machine learning, or worked on multiple AI systems.

Continuing with the first section concerning productivity, the results display overall satisfaction with the notation. The average and median result in three with a slight upwards skew towards higher likeliness. As a result, some productivity benefits were observed, but no significant increase was found. One interesting outlier would be the productivity, which has been experienced as unlikely to be boosted by about a third of all responses.Furthermore, when looking specifically at the quartiles and the interquartile range, the results can be considered consistent.

The picture is similar when it comes to perceived ease of use. Again, with a mean and median of three, there is a low level of agreement with most statements. An interesting observation is the 'negative' bias towards indifference. A closer look at the individual results shows that 'Ease of use' and 'Flexibility' in particular are marked as neither likely nor unlikely more often than most other results. Thus, two of the requirements that are considered to be core design focuses were not sufficiently met.

Concluding the interpretation of the questionnaire, the open questions are to be interpreted and condensed. Concerning the modelling experience, mostly standardized notation formats and pre-defined elements have been perceived as helpful and productive. Moving on, overall little insight on the provided separate workflow and information/detail attachment to individual elements has been provided, though it was mentioned as useful once. Conversely, other additions, like different processing elements, have been described as confusing.

Regarding additions and changes, more granular types, the addition of short description, and the ability to link individual elements to external sources as part of the notation have been mentioned. Furthermore, the addition of hiding some content was noted. Considering that a tutorial file has been created describing the purpose of existing elements and covering the option to extend the existing notation with custom elements, these results may indicate a need to improve the documentation and tutorials. With regard to the tutorial file, it was found to be inconsistent as it presented some additional elements that are not included in the standard template. Therefore, in combination with the results of the quantitative analysis, the feasibility of the proposed extension can be considered doubtful.

7. Discussion

With all the results presented, attention can be turned to answering the research questions of this thesis. The following sections provide the answers to each of the sub-questions RQ1 to RQ3, as well as a final answer to the overall research question. Finally, the limitations of this thesis will be stated to conclude this penultimate chapter of the thesis.

7.1. RQ1

Starting with the theoretical implications of RQ1: *”Who are the stakeholders, and what are their requirements for AI systems documentation?”*, numerous parties with varying interests in the documentation have been found. Thus said groups have been categories into different groups. One of arguably most requirement-heavy groups would be the *legal entities*. This group mainly harbours individuals who set policies that affect AI systems, especially executives who audit such systems and enforce legal bodies [KT21], [KT22], [HMD+23], [MZM21], [RRF+22], [RRF+22], [LWM24]. Another influential group are the *producers of AI systems*. Within this group, mainly three parties are having requirements on the documentation of AI systems. First, the developers and associated roles tasked with creating and maintaining such systems do have a range of requirements regarding documenting their system. In addition, there are subject matter experts who may have a better understanding of an AI system’s domain than the developers. Consequently, this domain knowledge has to be available in form of documentation. Then, there are the product owners, who have to understand the system in order to communicate effectively with other interested parties, the investors, as well as the internal audit, or quality assurance tasked with upholding quality standards and regulations [KT21], [KT22], [MHDSG23],[RRF+22], [PHB+18], [Gün20], [Mil22], [GJS22].

Moving on to the consumer perspective, *users* of the AI system present some relevant requirements. A user in this case does not just entail private persons, but also other companies for instance [HMD+23], [Gün20], [WBD+21], [RRF+22], [PHB+18], [Mil22]. Next is a group often depicted as distinct to to other groups presented thus far , namely the *academic sector*. The academic sector includes researches at various stages of their academic career, such as students or professors [GJS22], [PHB+18]. Other found stakeholders are the *bystanders*, who are indirectly affected by the usage of AI systems and thus have ”tacit” requirements [GJS22], [WBD+21], as well as the *individuals actively promoting ethical usage of AI* [PHB+18].

As for the requirement themselves, a plethora of different requirements could be detected. Initializing this topic, the first apparent set of requirements would be documenting all workflow aspects, which leads to requirements such as the description of the system itself, change management, design decisions, or description of the output [KT21],[HMD+23], [KHK+24], [LWM24]. Another set of findings relates to the inclusion of ecosystem perspectives in addition to the technical aspects of the documentation. Some examples include displaying risk management, description of the application domain, or descriptions about human roles and sources of error when interacting with the AI system [GJS22], [CDVR22],[WBD+21], [PHB+18], [MHDSG23].

Special emphasis was also set on including documentation on the data at multiple stages before usage. Specifically, where the data originates from, how the triage process of the data is conducted, what is done to prepare the data for usage, and how it is provided to the system were found to be relevant [KNHJ+23], [HMD+23], [WBD+21] [KT22], [GJS22]. Next, some needs were identified in relation to documentation standards. According to these requirements, there should be a documentation standard that includes the following best practice documentation guidelines and known standards [KT22], [CC22], [HMD+23], [LWM24]. Next, the documentation shall act as a ethical guide for making ethical sound decisions [BHKST22], [MHDSG23],[MZM21], [DYMY21], [PHB+18], [KT22], [LWM24]. Finally, there are some general non-functional requirements to consider, such as that the documentation is transparent, comprehensive, reproducible, easy to learn, explainable to all parties, and at the right level of granularity for each perspective [MHDSG23], [WBD+21], [LWM24], [CDVR22], [CC22], [KT21], [KT22], [HMD+23], [GJS22].

Commencing to the practical indication, it has been shown that a wide variety with a plethora of different perspectives are to be considered to answer this question properly. For further development, the requirements already provide some early insights into what needs have arisen, which can be investigated by the scope of the second research question. Taking the theoretical implications into consideration, the representation does have to have the capability of covering many perspectives comprehensibly. Doing so via a written description is certainly possible, but may also leave the room for ambiguous interpretation [KT21], [HMD+23], [PHB+18], [CDVR22], [CC22] [RRF+22]. Thus, other means, such as diagrams will likely be required. That being said, deciding on the right notation or the right combination of different diagrams may be a challenge on its own right. Resulting from this predicament, a tool or tools which have the option to create diagrams in multiple notations is also will to be a necessity. Ideally, it would also allow to directly link to other documents. However, this can also easily be achieved by either using a reference to the document location in a diagram, such as an URL, URI, or even local storage path. One interesting additional approach would be using other means not mentioned. Little to no literature has been found on using techniques, such as recorded audio and/or video for documentation. Thus, the remainder of the practical implications for the first research question will focus on diagrams from now on.

Directly addressing the found needs, other practical requirements in addition to a diagram editor and software to write textual descriptions would be options to show related content together, validation of the chosen notation, and potentially options to layer the diagram. Displaying related content together is interpreted as consequence from requirements which mention adding documentation on content typically not found workflow diagrams. Yet again, it would be beneficial to have a form to link the content to other sources, like URLs embedded into the diagram at the respective points of relevance. Validation per se may be a bit more challenging, as it is vastly dependent on the notation used and how easy or challenging it is to determine syntactic and semantic correctness. As such, this point has been omitted as strict requirement for the scope of this work. Finally, as far as the layers are concerned, this requirement stems mainly from the non-functional requirements. Most of these non-functional requirements aim at easy comprehension for different interested parties with low learning costs. As such, constructing diagrams in a modular form and having different levels of granularity would help to achieve these needs [MHDSG23], [WBD+21], [KT21], [CC22], [PHB+18].

7.2. RQ2

Armored with knowing the requirements, the second research question: *"How to represent AI system workflow to support AI systems documentation?"* can be tackled. Concerning theoretical implications, the research has shown, that either visual diagrams, textual description, or a combination thereof appear to be the best solution. It should also be noted that no definitive answer can be given as this research question encompasses a wide range of scenarios and therefore the answer will change. This includes the potential question of how many diagrams and/or written content should be used [KPL+14], [AF21].

In general, however, UML can cover most requirements if any type of UML model is considered. However, depending on the use case, this statement may be accompanied by multiple diagrams, making the result quite lengthy and potentially overwhelming. To some extent, TM modelling can achieve the same result if the scenario does not require too much detail and additional content. Similarly, depending on the layout, the flexibility of ontologies allows most workflow, ecosystem and data requirements to be covered. Furthermore, ethical requirements may also be feasible. However, following a potential documentation standard may prove challenging as the rules used to standardise ontologies would need to be consistent across different scenarios. Next, boxologies focus only on the workflow and therefore require many external resources to cover most requirements. However, it should be noted that the example found is specific to an AI system, so it may be easier to include additional content than for other notations. For data cards, the picture is very similar to boxologies. However, while data cards are more flexible in nature, they are not focused on AI systems and therefore need to be more tailored to such systems. Finally, Themisto undoubtedly has its use case in documenting development by automatically documenting code, but it does not help to cover AI systems in general.

As for the practical implications, the question on how to represent the AI workflow in order to cover all the requirements detected may deserve a work on its own. In order to stay within the scope of this paper, the following paragraphs will focus on which notation or combination of notations might be sufficient to answer this particular question, and in which scenario they will work best. The literature has shown various options to support AI system documentation. The candidates found most fitting the requirements from the first section include *Themisto* [WWD+22], *Ontologies* [NMEC21], [KPL+14], [ELS+23a], several *UML Diagrams* [MLNN20], [HG22], [JPZ22], [AF21], *Boxologies* [VHTT19], *Data Cards* [PZK22], as well as *TM Modeling* [AF21]. Each of these notations has their own benefits and drawbacks. Furthermore, the research has shown, that it is unlikely to find the right notation fitting every use case sufficiently, even within the same organization. Thus, the correct approach is dependent on the parameters of each particular scenario [KPL+14], [AF21]. Therefore, to answer this research question in a suiting manner, the requirements identified for each stakeholder group will be compared with the diagram types identified and suggestions on the most suiting options are made.

Starting withe the legal entities, which themselves have the most presence in the found literature, have also the widest ranging set of different requirements in both functional- and non functional requirements.Most of these requirements revolve around the presentation of the AI system, its capabilities and additional measures to meet legal requirements and allow for proper auditing. In particular, the risk assessment needs to be clearly presented in terms of accountability, harm to other parties and the potential for misuse. Finally, the documentation has to be understood uniformly [HMD+23], [MHDSG23], [KT21], [WBD+21], [BAW+20]. [MZM21].

Thus, extensive documentation will most likely be needed. For use cases resembling this approximate description, the recommendation would be to use a combination of an UML activity diagram, an UML use case diagram, data cards, Themisto, as well as an ontology. Each of the UML diagrams is recommended to represent the workflow, as well as the interaction with the diagram, in a well-established documentation standard with already established and generally recognised elements among experts . Thus, requirements such as displaying human actors, accountability, showing safety measures can be met [HG22], [JPZ22], [AF21], [EuA24c]. In addition, for audits in particular, including state charts and class diagrams may be helpful as well. The data cards are utilized to describe relevant additional information, such as the accuracy, bias and countermeasures in the data, or conclusions about the output [PZK22]. Themisto could be used for code documentation an particularly for the data collection and preparation process [WWD+22]. Finally, a basic ontology is used in an abstract fashion to link the content together [KPL+14], [SRMP+22].

Moving on to the producers, again different descriptions and most of the non-functional requirements were found to be necessary. Contrary to the legal entities, which can be deduced from the lack of evidence in the literature, the producers tend to focus more on the technical aspects and data requirements, adhering to standards for audits, and less on environmental impacts. With regard to data requirements, it is particularly interesting to note that there is little information in the literature found on the origin and preparation of the data [GJS22], [MZM21], [CDVR22], [KNHJ+23], [CC22], [BHKST22]. Given that most of the parameters match the parameters of the legal entities, generally the same setup could be recommended. However, bearing in mind that the producers include the developers, technical formats, such as state-charts, system sequence diagrams, and class diagrams could be beneficial as well. Should the AI system be considered non-risky, some additional steps, such as having use case diagrams to highlight risky interactions could be omitted.

Addressing the academic sector, most of the workflow- and ecosystem-requirements stay consistent to the former groups. That being said, there appears to be emphasis on describing the data origin and the cleaning process. Additionally, the documentation should be universally understandable, possibly even more than other stakeholder groups require, as results described should be reproducible [KHK$^+$24] [LWM24], [BML$^+$23], [DYMY21], [HMV$^+$22]. Given that there is no focus on audits and no strict requirement to include additional content, as shown in the legal environment, the setup can be an interesting combination, as the documentation of the workflow itself can be more flexible according to the records found, as long as it is easily understandable. Conversely, the documentation should be more rigorous about the input and output data. As a result, either an activity diagram, a system sequence diagram, ontologies or boxologies should be sufficient to document the workflow. Regarding data and output, data cards seem to be the most suitable option.

Next on the list are the users. In order to meet the requirements of the users, the main issues to be addressed are how the AI system works, its purpose, the data used and how data security is ensured. In addition, the documentation should be transparent and understandable. Finally, the measures on how the AI system is used in an ethically sound way are to be elaborated [HMD$^+$23], [GJS22], [KT22], [BHKST22]. Thus, to display the workflow, either boxologies or ontologies could be used. To facilitate elaboration on the data safety, especially for users not affine with technical concepts, either data cards or an additional ontology would be applicable.

For ethic activists almost all models could be utilized. As factors such as accountability and the description of certain design decisions is relevant, data cards appear the most promising result in addition to the general workflow [KT21]. For the workflow itself, an easy to learn option which is non-ambiguous would be best. Thus, options like boxologies may fit best. A similar assessment can be made for bystanders, as knowledge of the system and how it may affect them is a key concern.

7.3. RQ3

Finally, the question: *"How to design and develop tool supports for representing system workflow in the context of AI system documentation?"* can be addressed. As far as theoretical implications are concerned, automation in the broadest sense can be very effective in promoting effective documentation. There are many existing notations for creating diagrams to support documentation tasks. In addition, many allow for the creation of custom elements and predefined templates, increasing granularity and reusability. Even in a narrower sense, as shown in the practical implementations below, much of the documentation process can be automated. In general, most software allows machine interaction via APIs or similar means. As such, automation can be achieved via development of scripts and similar tools. In the context of this work, automation can be achieved by options such as extracting the information from a textual description or diagram and processing it further. Such extracted data can be used to create another type of data storage. An example of such a tool would be Chowlk, which converts existing Draw.io diagrams into an RDF format. [CFGCPV22]. Another idea would be to prompt the extracted information to tools created to generate documentation, like Themisto [WWD$^+$22].

Continuing, the remainder of this chapter will revolve around answering to what extend automation of AI system documentation can be supported via workflow representations of the system. To answer the aforementioned question, a proof of concept in form of the BEAM toolkit has been developed. As established before, it is challenging to meet all found requirements aptly. Thus, as an initial step, BEAM focuses on the requirements of the *academic sector* and *producers* specifically. BEAM consists of two main components, the notation and scripts that extract the content of diagrams modelled in this notation. Starting with the notation, there are two sources of evaluation. The first would be the requirements found in the literature and how they were met. The second source is a questionnaire that was carried out to collect initial feedback on the notation. To cover the comparison with the requirements from previous research, the notation is first compared with the requirements found for the selected stakeholder. This is followed by a comparison with the other requirements to provide a first check of suitability for other potentially interested parties.

But first, the requirements of researches and developers are to be summarized briefly. Initializing with the producers, many additional content is to be covered, apart from the overall system architecture [KHK+24]. To incorporate additional content, the option to attach further details and to leave comments have been introduced. While these options may not necessarily help to seamlessly integrate information, such as how the risk is managed in each step, they most certainly allow to attach links to other information sources, such as data cards. Furthermore, the ability to expand and collapse content, although specifically designed for Draw.io's mechanics, represents some steps towards greater transparency, adaptability and unambiguity. In terms of documentation standards, this point is difficult to judge at this stage, as the notation itself adapts to a currently existing notation and thus interferes with a standard, if the original notation is considered as such. But even so, the high degree of flexibility makes it difficult to comply with standards in general. Shifting attention to low learning costs to achieve this goal is not justifiable, as expansion involves learning new elements as well as new rules. Moreover, as mentioned before, the notation opts to provide the flexibility needed to cover as many different scenarios as possible. Consequently, low learning cost may not be achievable in many cases. Finally, there is also the requirement to incentives creating documentation. Again, this is not something that can be reasonably assessed. There are measures, such as pre-defined elements and templates, aimed at increasing productivity. It could be argued that this could also lead to an increase in the willingness to document. However, testing this hypothesis is beyond the scope of this paper.

Concerning the requirements of the academic sector, most points overlap with the producers. However, the entire process of obtaining and managing data is to be reflected on. Generally, the original boxology had elements covering actions around data processing. That being said, they were not designed to describe common necessary tasks, such as the data collection process. To aid with this problem, the BEAM notation again offers to attach details in form of comments and other elements to elements of the main workflow. However, no additional elements were introduced specifically for data collection and preprocessing. This design decision is mainly based on the idea of adding as few additional elements as possible in order to keep the additional learning costs low [KT21],[PHB+18].

Moving on to requirements from other stakeholders, much of the content developed for the previous requirements will be applicable in many other cases. Additional content, such as documentation of quality management, risk management, bias and its countermeasures, robustness, testing or relevance can be done by adding further details to the main workflow or by linking to another documentation artefact. One workflow requirement in particular remains to be addressed, namely the documentation of changes between versions. BEAM as a notation has no direct way of showing changes prominently. This could be done in an improvised way, however. One example of such a workaround would be to crate multiple layers and offset the elements that have changed to show the differences when both layers are displayed. That being said, it might be an interesting addition to incorporate such features into the notation itself via new elements.

The final point to consider for the notation extension is the questionnaire. In terms of the usefulness of the extended notation, there were mixed responses. Whilst it was generally appreciated that the library provides pre-defined elements to increase productivity and a legend to aid understanding of the model, some additional features such as lists were rarely mentioned in the responses. The ability to link additional details and to show or hide these details was little used overall. Nevertheless, it was seen as helpful, given some of the responses to the open questions. In addition, the processing element was found to be somewhat confusing and may need to be made more understandable. Another point to mention is the provided tutorials. Some elements were described, which are not part of the default template. This has lead to some confusion as well. As a final remark, the overall average performance in the perceived ease of use indicates an improvable learning cost for future development of the notation.

With the feedback covered, practical implications arising from the questionnaire are to be described briefly. In particular, most of the additional comprehensive elements were not used. This could be interpreted as an indication that this first attempt is not the right approach. In particular, the criticism of the open-ended responses has been taken into account in the following considerations. Firstly, as the option to introduce user-defined elements was not widely used, a possible future development of the BEAM notation could be to offer more elements without the option to introduce user-defined elements. Furthermore, using lists, which can be extended by further lists and encouraging using them as storage for a selection of elements to choose from has been confusing and not received too well. One way to improve this point may be to use a concept like UML class diagrams have introduced and detach these options to be an element residing on side of the page. The connection then is made by assigning the enumeration name as type to the element which shall have said options [Rum16]. Another idea would be to introduce a more granular layer with this option. However, this may lead to more model overhead and possibly comprehension issues. Next, the introduction of additional connector types is to be argued. While they may be necessary in some form from a technical point of view, they appear to be confusing rather than helpful. Therefore, for future reference, they could be removed and replaced by other indicators, such as another container containing the information and indicating the attachment of further information by naming the container accordingly. Finally, it should also be noted that the requirement to use the elements provided specifically, or to attach further properties to each new element, has also been criticised as a hindrance to productivity.

Many of these design decisions originate from a requirement arising from the scripts developed for this work. While the persons who have tried BEAM and provided feedback only had access to the notation, there is also the actual research question to consider to develop BEAM further. These scripts can use the extracted content to restructure it for further processing and check that the diagrams are syntactically correct (to some extent). However, since the ideal scenario would be to provide software that could work with non-ideal conditions (essentially ”wrong” diagrams), the code was not part of the testing phase for the questionnaire. The result has some interesting implications for the code. Based on the feedback, it is likely that many of the rules will not be followed, leading to unexpected results when converting the diagram to other formats. To solve this issue, a validator has been included as well. However, it will be near impossible to catch every error and ask the user for correction. Thus, if the automatic extraction and error checking is pursuit further, some aspects of the code are likely to be improved as well.One of the most annoying issues may be that many elements do not have the appropriate properties assigned to them, as this is already considered obstructive already. Thus, extending the extraction by not looking for the ”element” property, but also at the label and shape and assuming that these properties are correct may be a viable option. Alternatively, the notation could be changed to remove the requirement of a custom property and just operate based on label and shape. The latter option would also simplify to extraction process of the received XML files, but might also increase the potential of typos and again. Another potential issue would be the concern of standardization. Currently, only the checks for the elements of the original boxology are statically stored in a configuration file. Other checks rely on the extracted legend. If only standardised elements are to be used, there is currently no way to enforce this behaviour.

Moving on to other modules that have also been developed in order to gain more insight into answering the final research question. Said modules entail a small web-application as well as the querying module show that there are multiple applications for automation. The web-application shall show a concept on how a complete product could be offered in a public domain. Conversely, the query module will show how one or more files stored in restructured format can be searched for their content easily and quickly. In summary, a well-established notation with the right editor can help automate the documentation of AI systems in a variety of ways, including remodelling, content validation, and content retrieval. Furthermore, these are just a few examples of use cases. In the future, ideas such as querying an Large Language Model (LLM) for information about a diagram to generate concise textual descriptions and possibly diagrams in other notations could be further applications [WWD+22].

7.4. Main Research Question

With all the sub-questions covered, the main research question: "*To what extent can the automation of AI systems documentation be supported through system workflow representation?*" can be answered. As the content has shown, the question itself can be answered rather directly, but it contains many facets. In principle, as long as the data in the diagram is available, virtually anything could be done to display, store, transform or manipulate the content. This thesis has demonstrated this by developing prototypes for extracting the information from diagrams, transforming it into other structures, validating the data, and allowing the data to be queried. In addition, a measure has been introduced to show how these tools could be offered outside of development in the form of a small web -application.

However, in order to carry out all these processes, the data on a diagram must be interpreted correctly. This requires a standardised way of reading the input. While there are many options that would work to some extent, none of them would meet all the requirements found in the literature. Therefore, an extension of a boxology has been introduced to facilitate the requirements of at least two stakeholders [VHTT19]. This extension is an attempt to better meet these requirements and to extend the range of automation options.

7.5. Limitations

As with most scientific work, it is unfortunately impossible to cover all the content of the field that may be relevant to some of the content of this particular work. In addition, time and resource constraints have to be taken into account. Consequently, in order to remain within these constraints, some significant limitations have had to be made.

First, the selection of literature is to be illustrated. While most found literature may contain relevant insights, it is impossible to cover all the content sufficiently. Thus, especially for the found notations and editors, if little to no evidence has been found in their application on AI systems, they have been excluded. Furthermore, this work does not include Explainable AI (XAI) as one of its topics, although it is undoubtedly relevant to advancing the field of documentation of AI and AI systems. This decision is based on the difference in scope. Traditionally, XAI focuses on the detailed elaboration of a particular AI model. In addition, many of the requirements found, such as the elaboration of bias and bias detection methods, ultimately focus on explaining components of the AI model in more detail. However, these requirements are included as they remain relevant on a larger scale. Nevertheless, as the premise of this work is AI systems, XAI has not been explored in more detail. [GJS22].

Continuing, the criteria for comparing the diagramming tools are to be elaborated briefly. As the matter of choosing the right editor is not strictly bound to science, some non-scientific sources, as well as small degree subjective judgement based on the available resources for the development have been used. Specifically the requirement of open-source and or freely available tools falls under the limitation of resources. Other requirements, such as having import- and export-options arise from the requirement further process it with developed scripts. While this scenario may not be the same for other works, they had to be considered to create this work.

Another limitation is the timing of this work. While the inclusion of documentation standards such as those required by the EU AI Act would be invaluable to this work, to the knowledge of the author they have not yet been developed [EuA24a]. Therefore, while they may be a source to consider in the near future, this work must be considered under the assumption that there is no current standard for the documentation of AI systems in particular.

In terms of the content developed, there are also a number of limitations to be considered. This work provides a proof of concept of what a notation for AI systems might look like and how such a notation, once established, could be used for further automation in terms of documentation. It is not the objective of this work to provide a fully functional version of the final product. Furthermore, as far as the notation itself is concerned, the notation is primarily tailored to a specific set of requirements. Although an attempt has been made to cover all requirements found, only the selected few requirements have been considered "significant" for the time being. Thus, although these requirements may not part of this work, future adaptations may focus on other requirements and other scenarios.

8. Conclusion & Future Outlook

Creating documentation for AI systems, which is unambiguous and transparent for every party reading it remains a challenge up to this day. This work aims to tackle some of these challenges and provide insights on multiple issues of the field. First, the actual target audience using the documentation and their requirements had to be defined. This objective was met by conducting a literature research of recent studies covering requirements of documentation regarding AI systems or hybrid systems [KT21], [Mil22].

Based on this result, the next goal was to detect notations which are compatible to meet as many of the uncovered requirements as possible. The result of this section unveiled into a wide selection of different well-known and widely used diagram types, which have been used to represent AI systems. The most popular examples entail UML diagrams like activity diagrams, state charts, use case diagrams, system sequence diagrams, or other extensions based on UML, like TM Modeling [MLNN20], [HG22], [JPZ22], [AF21], [KPL^{+}14].

Furthermore, some works tended to use more flexible options, such as ontolgies and data cards [NMEC21], [SRMP^{+}22], [DG23], [PHB^{+}18], [KPL^{+}14], Finally, the last "competitor" would be a notation bases on a boxology specifically designed to design AI- and hybrid systems [VHTT19]. Each of the aforementioned styles had their own benefits of drawbacks. Crudely speaking, more static options, such as UML diagrams, are considered to be well-established and familiar for software documentation and may therefore require little additional learning. However, some scenarios have shown that UML diagrams may not be able to adequately cover all content due to the pre-defined elements. For example, data flows in particular can be difficult to represent without combining several diagrams [MZM21],[HG22]. Conversely, more flexible options, such as ontologies or data cards allow to incorporate scenarios as deemed necessary for the situation. However, this additional flexibility comes at the price of leading to inconsistencies. If no standard is followed by convention, possibly transparency- as well as comprehension-troubles might occur [NMEC21], [PZK22].

Thus, there is no jack of all trades. Most of the presented methods allow to to cover most of the content requirements to a certain extent. However, to cover all necessities, a combination of multiple diagrams may be needed. That being said, when using multiple diagrams to accurately represent the system, other factors need to be considered, such as inconsistencies between different models, keeping the content concise so as not to overwhelm the reader, or the additional learning costs required to understand each different diagram type. Thus, some approaches have opted to combine different models. However, this option might lead to complex and hardly understandable models at scale again [KPL^{+}14], [AF21].

Equipped with this knowledge, the final part of this work is to tackle some of the requirements currently not met sufficiently in addition to determine to what extend automation could help AI system workflow documentation. To achieve this, the producers and the academic sector have been chosen as the starting point. The objective is to meet the requirements previously not met with other notations. Thus, to meet these requirements, an extension of the presented boxology called *BEAM* has been proposed [VHTT19]. BEAM adds some additional elements that allow for more flexible options in terms of adding relevant content, as well as expandable options to include more granular information. In addition, BEAM invites the introduction of more elements to the diagram and the modification of existing elements as deemed necessary. In order to maintain an overview of the current AI system being represented, BEAM also requires a legend that includes the shape of the element and an appropriate label. To collect initial feedback on BEAM, a survey based on the TAM model has been conducted [AEMK18]. The results showed some mixed feelings about the BEAM notation. Generally, having a legend was considered an improvement. In addition, the ability to attach comments also seems to be well received. However, the option to extend to different levels of granularity and generally introducing or altering elements has rarely been used and was not considered too helpful. In addition, some elements, such as the use of lists or different types of processes, were confusing in some instances.

Finally, to check the extent of possible automation, some scripts have been developed and compiled to a toolkit as a proof of concept. These scripts allow to extract information from a model modelled in BEAM and provided in an XML format, to restructure its content and to validate some of the syntax and semantics. Additionally, to allow BEAM to be used by a broader audience, a small Flask app has been written [Gri18]. Finally, another selectable mode has been introduced which allows one or more restructured BEAM models to be queried for content.

For future works who may want to collect further insight, some initial directions might be suggestible. Starting with the stakeholders and requirements, this work has focused on detecting all interested stakeholders and their requirements. However, the importance and a ranking of the importance of different requirements has not been covered to the knowledge of the author. Thus, some approaches for future works would be to check how important the different requirements are for the respective stakeholders. Additionally, as discussed at multiple points throughout this work, the importance of requirements is also highly dependent on context. Thus another direction would be to check the importance of the same requirements in different use cases. In addition, as briefly indicated in the discussion, different approach to current or new notations may better meet the needs of other stakeholders.

As far as future practical work is concerned, the only evidence found during the research for this thesis was the use of written documents and diagrams. However, another approach may be to also incorporate audio and video as a form of documentation. Alternatively, the proposed notation may also be altered and extended based on the feedback collected. In terms of the developed scripts, they contain a very crude first proof of concept and can be improved and extended in many ways. Firstly, the extraction process may be optimized by also allowing other options, such as multiple diagrams or different formats. The same statement is true for the restructuring process. For example, automatic restructuring into and RDF format can be envisaged. Next, the web-application can be extended to offer via a fully functioning website. With the web-application, other options such as displaying the result of the restructuring and validation can be offered directly on the website for live customisation. Finally, the querying module can be extended to allow for more sophisticated queries. One final, more resource-intensive addition would be to adapt a more suitable option for creating or editing diagrams, or alternatively develop a custom diagramming tool specifically targeted at AI system notations or BEAM in particular. As indicated in this work, validation of the syntax and semantic of diagrams in a specific notation is a challenge, which may be worth its own project. To do so, a more specialised approach focusing on BEAM alone, or by extension on other AI system notations would help to further advance the goals of a low cost of learning and transparency.Finally, in order to provide a refined version for testing, it may be helpful to create a full product from the BEAM notation and prototype toolkit presented in this thesis.

Appendix

A. Literature Research Results

The following table shows the literature found that was considered relevant to the content of the literature review in order to answer the respective research questions.

Article	Key Phrase Used	Database	Last Accessed	Reference
Experiences with improving the transparency of AI models and services	Stakeholder of AI documentation	Google scholar	25.04.2024	[HHM+20]
How ai developers overcome communication challenges in a multidisciplinary team: A case study	Stakeholder of AI documentation	Google Scholar	24.04.2025	[PPW+21]
Toward trustworthy AI development: mechanisms for supporting verifiable claims	Stakeholder of AI documentation	Google Scholar	24.04.2025	[BAW+20]
Documenting high-risk AI: a European regulatory perspective	Stakeholder of AI documentation	Google Scholar	27.04.2025	[HMD+23]
Creating Value with Artificial Intelligence: A Multi-stakeholder Perspective	Stakeholder AI	Google Scholar	18.04.2024	[Gün20]
Stakeholders in explainable AI	Stakeholder AI; Stakeholder - AI systems	Google Scholar	15.04.2024	[PHB+18]
Software documentation is not enough! Requirements for the documentation of AI	Documentation of AI, standards of AI documentation; Requirements of AI documentation	Emerald; Google Scholar	22.04.2024	[KT21]

AI Documentation: A path to accountability	Documentation of AI; Standards of AI documentation	Google Scholar; Elsevier	23.04.2024	[KT22]
Towards ecosystems for responsible AI: expectations on sociotechnical systems	Documentation of AI	Google Scholar	17.04.2024	[MZM21]
Provenance documentation to enable explainable and trustworthy AI: A literature review	Standards of AI documentation	Google Scholar	17.04.2024	[KNHJ+23]
Open Datasheets: Machine-readable Documentation for Open Datasets and Responsible AI Assessments	Standards of AI documentation	Google Scholar	19.04.2024	[RVS+23]
The sanction of authority: Promoting public trust in AI	Standards of AI documentation	Google Scholar	19.04.2024	[KR21]
The landscape of data and AI documentation approaches in the European policy context	Standards of AI documentation	Google Scholar	20.04.2024	[MHDSG23]
Interactive model cards: A human-centered approach to model documentation	Standards of AI documentation	Google Scholar	21.04.2024	[CDVR22]
Black Box or Open Science? A Study On Reproducibility In AI Development Papers	Standards of AI documentation	Google Scholar	21.04.2024	[KHK+24]
Data cards: Purposeful and transparent dataset documentation for responsible ai	Standards of AI documentation; Requirements of AI documentation	Google Scholar	22.04.2024	[PZK22]

The role of artificial intelligence model documentation in translational science: scoping review	Standards of AI documentation	Google Scholar	16.04.2024	[BML+23]
Understanding implementation challenges in machine learning documentation	Requirements of AI documentation	Google Scholar	16.04.2024	[CC22]
Use case cards: A use case reporting framework inspired by the european AI act	Requirements of AI documentation	Google Scholar	16.04.2024	[HFLBG24]
Understanding machine learning practitioners' data documentation perceptions, needs, challenges, and desiderata	Requirements of AI documentation	Google Scholar	16.04.2024	[HMV+22]
Ai transparency in the age of llms: A human-centered research roadmap	Requirements of AI documentation	Google Scholar	16.04.2024	[LV23]
P7001: A proposed standard on transparency	Requirements of AI documentation	Google Scholar	18.04.2024	[WBD+21]
Educating software and AI stakeholders about algorithmic fairness, accountability, transparency and ethics.	Requirements of AI documentation	Google Scholar	26.04.2024	[BHKST22]
Explainable AI, but explainable to whom? An exploratory case study of xAI in healthcare	Requirements of AI documentation	Google Scholar	26.04.2024	[GJS22]
AI lifecycle models need to be revised: An exploratory study in Fintech	Requirements of AI documentation	Google Scholar	28.04.2024	[GJS22]

Towards personalized explanations for AI systems: designing a role model for explainable AI in auditing	Stakeholder - AI systems	Google Scholar	18.07.2024	[RRF$^+$22]
Stakeholder Participation in AI: Beyond" Add Diverse Stakeholders and Stir"	Stakeholder - AI systems	Google Scholar	05.04.2024	[DYMY21]
Stakeholder roles in artificial intelligence projects	Stakeholder - AI documentation	Google Scholar	04.04.2024	[Mil22]
Three pathways for standardisation and ethical disclosure by default under the European Union Artificial Intelligence Act	Stakeholder of AI documentation	Google Scopus	08.04.2024	[LWM24]
Documentation matters: Human-cantered ai system to assist data science code documentation in computational notebooks	AI system workflow documentation	Google Scholar	28.03.2024	[WWD$^+$22]
A semantic framework to support AI system accountability and audit	AI system workflow documentation	Google Scholar	29.03.2024	[NMEC21]
Lynx: A knowledge-based AI service platform for content processing, enrichment and analysis for the legal domain	AI system workflow documentation; AI workflow documentation	Google Scholar	29.03.2024	[SRMP$^+$22]
The algorithm journey map: a tangible approach to implementing AI solutions in healthcare	AI system workflow documentation	Google Scholar	25.03.2024	[BHK$^+$24]

Data journeys: explaining ai workflows through abstraction	AI system workflow representation	Google Scholar	05.04.2024	[DG23]
Combining UML and ontology: An exploratory survey	UML for AI systems	Google Scholar	05.04.2024	[MLNN20]
Documenting use cases in the affective computing domain using Unified Modeling Language	UML for AI systems	Google Scholar	05.04.2024	[HG22]
Using the Strategy Design Pattern for Hybrid AI System Design	UML for AI systems	Google Scholar	06.04.2024	[JPZ22]
Ontology-based knowledge representation of industrial production workflow	workflow representation methods	Google Scholar	08.04.2024	[YZT+23]
Describing and organizing semantic web and machine learning systems in the swemls-kg	SWeMLS ontology for ai system	Google scholar	24.05.2024	[ELS+23a]
An ontology-based approach towards comprehensive workflow	Ontologies for workflow	Google Scholar	06.04.2024	[KPL+14]
Are use case and class diagrams complementary in requirements analysis? An experimental study on use case and class diagrams in UML	UML for AI systems	Google Scholar	08.04.2024	[SL04]

Table A.1.: Result of the Literature Research

B. Questionnaire

This section contains the questionnaire used to evaluate the BEAM notation extension.

B.1. Section A: Perceived Usefulness

Options to choose -> 1 = Extremely Likely, 2 = Quite likely 3 = Likely, 4 = Neither, 5 = Unlikely, 6 = Quite unlikely, 7 = Extremely unlikely

- **A001 - Efficiency & Communication:** Using Beam for documenting AI systems would enable me to document and communicate my project more efficiently.
- **A002 - Documentation Quality:** Beam would improve my raise the quality of my documentation.
- **A003 - Productivity:** Using Beam would increase my productivity.
- **A004 - Documentation Comprehension:** Using Beam would make my documentation more understandable.
- **A005 - Ease of Use:** Using Beam would make it easier to document AI systems.
- **A006 - Usefulness:** I would find Beam useful as a documentation tool.
- **A007 - Team Coordination:** Using Beam throughout development would improve coordination within the team.
- **A008 - Communication Stakeholders:** Using Beam would improve communication with my stakeholders.

B.2. Section B: Perceived Ease of Use

Options to choose -> 1 = Extremely Likely, 2 = Quite likely 3 = likely, 4 = neither, 5 = unlikely, 6 = quite unlikely, 7 = extremely unlikely

- **A101 - Beam vs. Text Description:** Learning to operate Beam would be easier for me than describing my AI system in text.
- **A102 - Ease of Use:** I would find it easy to get Beam to do what I want it to do.
- **A103 - Intuitiveness:** Beam is intuitive to use.
- **A104 - Flexibility:** Using Beam gives me the flexibility I need to document my AI system in an understandable way.
- **A105 - Custom Extensions:** It would be easy for me to adapt or extend the standard library provided if necessary.
- **A106 - Handling:** Beam is generally easy to use.

B.3. Section C: Open questions

- **A201 - Modelling Experience:** How did your modelling experience differ from previous manually created diagrams (if you have modelled any diagram before)?
- **A202 - Addition or Changes:** What changes or additions would you like to see in the Beam library?

B.4. Section D: Demographics

- **A301 - Modelling Experience:** What previous experience do you have in modelling AI systems?
- **A302 - Experience ML:** How much experience do you have in working with ML models?
- **A303 - Role:** Please select your current profession.

Options to choose:

A301 - Modelling Experience -> 1 = None, 2 = Some experience, 3 = Considerable experience

A302 - Experience Machine Learning -> 1 = Foundational knowledge, 2 = Worked on some introductory projects involving AI, 3 = Worked with multiple AI systems, 4 = Considerable experience

A303 - Roles -> 1 = Not occupying role, 2 = Occupying role

Bibliography

[AEMK18] M. Al-Emran, V. Mezhuyev, and A. Kamaludin, “Technology acceptance model in m-learning context: A systematic review,” *Computers & Education*, vol. 125, pp. 389–412, 2018.

[AF21] S. Al-Fedaghi, “Model multiplicity (uml) versus model singularity in system requirements and design,” *arXiv preprint arXiv:2105.00616*, 2021.

[Apa23] Apache, https://www.openoffice.org/product/draw.html, 2023, accessed on 2024-03-01.

[Atl24] Atlassian, “confluence,” https://www.atlassian.com/de/software/confluence, 2024.

[BAW$^+$20] M. Brundage, S. Avin, J. Wang, H. Belfield, G. Krueger, G. Hadfield, H. Khlaaf, J. Yang, H. Toner, R. Fong *et al.*, “Toward trustworthy ai development: mechanisms for supporting verifiable claims,” *arXiv preprint arXiv:2004.07213*, 2020.

[BHK$^+$24] W. Boag, A. Hasan, J. Y. Kim, M. Revoir, M. Nichols, W. Ratliff, M. Gao, S. Zilberstein, Z. Samad, Z. Hoodbhoy *et al.*, “The algorithm journey map: a tangible approach to implementing ai solutions in healthcare,” *npj Digital Medicine*, vol. 7, no. 1, p. 87, 2024.

[BHKST22] V. Bogina, A. Hartman, T. Kuflik, and A. Shulner-Tal, “Educating software and ai stakeholders about algorithmic fairness, accountability, transparency and ethics,” *International Journal of Artificial Intelligence in Education*, pp. 1–26, 2022.

[BML$^+$23] T. A. Brereton, M. M. Malik, M. Lifson, J. D. Greenwood, K. J. Peterson, and S. M. Overgaard, “The role of artificial intelligence model documentation in translational science: scoping review,” *Interactive Journal of Medical Research*, vol. 12, no. 1, p. e45903, 2023.

[CBB$^+$02] P. C. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord, and J. Stafford, “A practical method for documenting software architectures,” 2002.

[CC22] J. Chang and C. Custis, “Understanding implementation challenges in machine learning documentation,” in *Proceedings of the 2nd ACM Conference* *on Equity and Access in Algorithms, Mechanisms, and Optimization*, 2022, pp. 1–8.

[CDVR22] A. Crisan, M. Drouhard, J. Vig, and N. Rajani, "Interactive model cards: A human-centered approach to model documentation," in *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, 2022, pp. 427–439.

[CFGCPV22] S. Chávez-Feria, R. García-Castro, and M. Poveda-Villalón, "Chowlk: from uml-based ontology conceptualizations to owl," in *European Semantic Web Conference.* Springer, 2022, pp. 338–352.

[Cre24] Creately, https://creately.com/plans/, 2024, cinergix Pty Ltd, Accessed on 2024-03-02.

[CT12] M. Chinosi and A. Trombetta, "Bpmn: An introduction to the standard," *Computer Standards & Interfaces*, vol. 34, no. 1, pp. 124–134, 2012.

[CWG+21] H. Chen, Y. Wang, T. Guo, C. Xu, Y. Deng, Z. Liu, S. Ma, C. Xu, C. Xu, and W. Gao, "Pre-trained image processing transformer," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 12 299–12 310.

[DD24] Drawio-Desktop, https://github.com/jgraph/drawio-desktop/releases/tag/v23.1.5, 2024, accessed at 2024-03-03.

[DG23] E. Daga and P. Groth, "Data journeys: explaining ai workflows through abstraction," *Semantic Web*, no. Preprint, pp. 1–27, 2023.

[DR20] A. Das and P. Rad, "Opportunities and challenges in explainable artificial intelligence (xai): A survey," *arXiv preprint arXiv:2006.11371*, 2020.

[Dra23] Draw.io, "Draw.io website," https://www.drawio.com/, 2023, jGraph Ltd, Accessed on 2024-03-02.

[DTH01] M. Dumas and A. H. Ter Hofstede, "Uml activity diagrams as a workflow specification language," in *International conference on the unified modeling language.* Springer, 2001, pp. 76–90.

[DYMY21] F. Delgado, S. Yang, M. Madaio, and Q. Yang, "Stakeholder participation in ai: Beyond" add diverse stakeholders and stir"," *arXiv preprint arXiv:2111.01122*, 2021.

[ELS+23a] F. J. Ekaputra, M. Llugiqi, M. Sabou, A. Ekelhart, H. Paulheim, A. Breit, A. Revenko, L. Waltersdorfer, K. E. Farfar, and S. Auer, "Describing and organizing semantic web and machine learning systems in the swemls-kg," in *European Semantic Web Conference.* Springer, 2023, pp. 372–389.

[Els23b] Elsevier.com, "Scopus abstract and citation database," https://www.elsevier.com/products/scopus, 2023.

[Eme19] Emerald.com, "Emerald insight," https://www.emerald.com/insight/, 2019.

[EuA24a] "Eu ai act," https://www.euaiact.com/article/1, 4 2024, accessed on 2024-04-16.

[EuA24b] "Eu ai act," https://www.euaiact.com/article/3, 4 2024, accessed on 2024-04-25.

[EuA24c] "Eu ai act article 18," https://www.euaiact.com/article/18, 4 2024, accessed on 2024-04-16.

[git20] github, "Github," 2020. [Online]. Available: https://github.com/

[GJS22] J. Gerlings, M. S. Jensen, and A. Shollo, "Explainable ai, but explainable to whom? an exploratory case study of xai in healthcare," *Handbook of Artificial Intelligence in Healthcare: Vol 2: Practicalities and Prospects*, pp. 169–198, 2022.

[Gli07] M. Glinz, "On non-functional requirements," in *15th IEEE international requirements engineering conference (RE 2007)*. IEEE, 2007, pp. 21–26.

[Goo24] Google.com, "Google scholar," https://scholar.google.com/, 2024.

[Gri18] M. Grinberg, *Flask web development: developing web applications with python.* " O'Reilly Media, Inc.", 2018.

[Gün20] H. Güngör, "Creating value with artificial intelligence: A multi-stakeholder perspective," *Journal of Creating Value*, vol. 6, no. 1, pp. 72–85, 2020.

[HCHVD21] M. Haakman, L. Cruz, H. Huijgens, and A. Van Deursen, "Ai lifecycle models need to be revised: An exploratory study in fintech," *Empirical Software Engineering*, vol. 26, no. 5, p. 95, 2021.

[HFLBG24] I. Hupont, D. Fernández-Llorca, S. Baldassarri, and E. Gómez, "Use case cards: A use case reporting framework inspired by the european ai act," *Ethics and Information Technology*, vol. 26, no. 2, p. 19, 2024.

[HG22] I. Hupont and E. Gómez, "Documenting use cases in the affective computing domain using unified modeling language," in *2022 10th international conference on affective computing and intelligent interaction (ACII)*. IEEE, 2022, pp. 1–8.

[HHM$^+$20] M. Hind, S. Houde, J. Martino, A. Mojsilovic, D. Piorkowski, J. Richards, and K. R. Varshney, "Experiences with improving the transparency of ai models and services," in *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–8.

[HMD$^+$23] I. Hupont, M. Micheli, B. Delipetrev, E. Gómez, and J. S. Garrido, "Documenting high-risk ai: a european regulatory perspective," *Computer*, vol. 56, no. 5, pp. 18–27, 2023.

[HMV$^+$22] A. K. Heger, L. B. Marquis, M. Vorvoreanu, H. Wallach, and J. Wortman Vaughan, "Understanding machine learning practitioners' data documentation perceptions, needs, challenges, and desiderata," *Proceedings of* *the ACM on Human-Computer Interaction*, vol. 6, no. CSCW2, pp. 1–29, 2022.

[JPZ22] S. Jüngling, M. Peraic, and C. Zhu, "Using the strategy design pattern for hybrid ai system design." in *AAAI Spring Symposium: MAKE*, 2022.

[KEBP21] H. Koç, A. M. Erdoğan, Y. Barjakly, and S. Peker, "Uml diagrams in software engineering research: a systematic literature review," in *Proceedings*, vol. 74, no. 1. MDPI, 2021, p. 13.

[KHK$^+$24] F. Koenigstorfer, A. Haberl, D. Kowald, T. Ross-Hellauer, and S. Thalmann, "Black box or open science? assessing reproducibility-related documentation in ai research," 2024.

[KM02] A. Knapp and S. Merz, "Model checking and code generation for uml state machines and collaborations," 2002.

[KNHJ$^+$23] A. Kale, T. Nguyen, F. C. Harris Jr, C. Li, J. Zhang, and X. Ma, "Provenance documentation to enable explainable and trustworthy ai: A literature review," *Data Intelligence*, vol. 5, no. 1, pp. 139–162, 2023.

[Kni23] Knime, https://www.knime.com/, 2023, accessed at 2024-03-08.

[KPL$^+$14] M. N. Koukovini, E. I. Papagiannakopoulou, G. V. Lioudakis, N. Dellas, D. I. Kaklamani, and I. S. Venieris, "An ontology-based approach towards comprehensive workflow modelling," *IET software*, vol. 8, no. 2, pp. 73–85, 2014.

[KR21] B. Knowles and J. T. Richards, "The sanction of authority: Promoting public trust in ai," in *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, 2021, pp. 262–271.

[KT21] F. Königstorfer and S. Thalmann, "Software documentation is not enough! requirements for the documentation of ai," *Digital Policy, Regulation and Governance*, vol. 23, no. 5, pp. 475–488, 2021.

[KT22] F. Koenigstorfer and S. Thalmann, "Ai documentation: A path to accountability," *Journal of Responsible Technology*, vol. 11, p. 100043, 2022.

[LV23] Q. V. Liao and J. W. Vaughan, "Ai transparency in the age of llms: A human-centered research roadmap," *arXiv preprint arXiv:2306.01941*, 2023.

[LWM24] J. Laux, S. Wachter, and B. Mittelstadt, "Three pathways for standardisation and ethical disclosure by default under the european union artificial intelligence act," *Computer Law & Security Review*, vol. 53, p. 105957, 2024.

[MHDSG23] M. Micheli, I. Hupont, B. Delipetrev, and J. Soler-Garrido, "The landscape of data and ai documentation approaches in the european policy context," *Ethics and Information Technology*, vol. 25, no. 4, p. 56, 2023.

[Mil22] G. J. Miller, "Stakeholder roles in artificial intelligence projects," *Project Leadership and Society*, vol. 3, p. 100068, 2022.

[MLNN20] M. M. Mkhinini, O. Labbani-Narsis, and C. Nicolle, "Combining uml and ontology: An exploratory survey," *Computer Science Review*, vol. 35, p. 100223, 2020.

[Mur23] Mural, https://www.mural.co, 2023, lUMA Institute, LLC, Accessed on 2024-03-02.

[MZM21] M. Minkkinen, M. P. Zimmer, and M. Mäntymäki, "Towards ecosystems for responsible ai: expectations on sociotechnical systems, agendas, and networks in eu documents," in *Conference on e-Business, e-Services and e-Society.* Springer, 2021, pp. 220–232.

[NMEC21] I. Naja, M. Markovic, P. Edwards, and C. Cottrill, "A semantic framework to support ai system accountability and audit," in *The Semantic Web: 18th International Conference, ESWC 2021, Virtual Event, June 6–10, 2021, Proceedings 18.* Springer, 2021, pp. 160–176.

[PHB$^+$18] A. Preece, D. Harborne, D. Braines, R. Tomsett, and S. Chakraborty, "Stakeholders in explainable ai," *arXiv preprint arXiv:1810.00184*, 2018.

[PPW$^+$21] D. Piorkowski, S. Park, A. Y. Wang, D. Wang, M. Muller, and F. Portnoy, "How ai developers overcome communication challenges in a multidisciplinary team: A case study," *Proceedings of the ACM on Human-Computer Interaction*, vol. 5, no. CSCW1, pp. 1–25, 2021.

[PZK22] M. Pushkarna, A. Zaldivar, and O. Kjartansson, "Data cards: Purposeful and transparent dataset documentation for responsible ai," in *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, 2022, pp. 1776–1826.

[RRF$^+$22] J. Rebstadt, F. Remark, P. Fukas, P. Meier, and O. Thomas, "Towards personalized explanations for ai systems: designing a role model for explainable ai in auditing," 2022.

[RS04] J. Rowley and F. Slack, "Conducting a literature review," *Management research news*, vol. 27, no. 6, pp. 31–39, 2004.

[Rum16] B. Rumpe, *Modeling with UML.* Springer, 2016.

[RVS$^+$23] A. C. Roman, J. W. Vaughan, V. See, S. Ballard, N. Schifano, J. Torres, C. Robinson, and J. M. L. Ferres, "Open datasheets: Machine-readable documentation for open datasets and responsible ai assessments," *arXiv preprint arXiv:2312.06153*, 2023.

[SL04] K. Siau and L. Lee, "Are use case and class diagrams complementary in requirements analysis? an experimental study on use case and class diagrams in uml," *Requirements engineering*, vol. 9, pp. 229–237, 2004.

[SRMP$^+$22] J. M. Schneider, G. Rehm, E. Montiel-Ponsoda, V. Rodríguez-Doncel, P. Martín-Chozas, M. Navas-Loro, M. Kaltenböck, A. Revenko, S. Karampatakis, C. Sageder *et al.*, "Lynx: A knowledge-based ai service platform for content processing, enrichment and analysis for the legal domain," *Information Systems*, vol. 106, p. 101966, 2022.

[VBHM20] J. Vom Brocke, A. Hevner, and A. Maedche, "Introduction to design science research," *Design science research. Cases*, pp. 1–13, 2020.

[vdML02] T. von der Maßen and H. Lichter, "Modeling variability by uml use case diagrams," in *Proceedings of the International Workshop on Requirements Engineering for product lines*, 2002, pp. 19–25.

[VHTT19] F. Van Harmelen and A. Ten Teije, "A boxology of design patterns for hybrid learning and reasoning systems," *Journal of Web Engineering*, vol. 18, no. 1-3, pp. 97–123, 2019.

[VRDJ95] G. Van Rossum and F. L. Drake Jr, *Python reference manual.* Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[WBD$^+$21] A. F. Winfield, S. Booth, L. A. Dennis, T. Egawa, H. Hastie, N. Jacobs, R. I. Muttram, J. I. Olszewska, F. Rajabiyazdi, A. Theodorou *et al.*, "Ieee p7001: A proposed standard on transparency," *Frontiers in Robotics and AI*, vol. 8, p. 665729, 2021.

[Wu.23] Wu.ac.at, "Wu library," https://katalog.wu.ac.at/, 2023.

[WWD$^+$22] A. Y. Wang, D. Wang, J. Drozdal, M. Muller, S. Park, J. D. Weisz, X. Liu, L. Wu, and C. Dugan, "Documentation matters: Human-centered ai system to assist data science code documentation in computational notebooks," *ACM Transactions on Computer-Human Interaction*, vol. 29, no. 2, pp. 1–33, 2022.

[YZT$^+$23] C. Yang, Y. Zheng, X. Tu, R. Ala-Laurinaho, J. Autiosalo, O. Seppänen, and K. Tammi, "Ontology-based knowledge representation of industrial production workflow," *Advanced Engineering Informatics*, vol. 58, p. 102185, 2023.