ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

Addressing Fairness in AI Systems: Design and Development of a Pragmatic (Meta-)Methodology

Tesi di laurea in:
Intelligent Systems Engineering

Relatore
Prof. Giovanni Ciatto

Candidato
Mattia Matteini

Correlatori
Prof.ssa Roberta Calegari
Prof. Andrea Omicini

IV Sessione di Laurea
Anno Accademico 2023–2024

Abstract

Biases and discriminations are present in several Artificial Intelligence (AI) systems as much as they are rooted in the society. Fairness in AI refers to the development of software systems that do not exhibit biases or systematic discriminations against specific individuals or groups. Addressing fairness is particularly challenging because it requires balancing ethical, social, legal, and technical expertises. This thesis proposes a meta-methodology for building fair AI systems, offering both a conceptual framework and a concrete software tool implementing the methodology. Instead of a single solution for all kinds of AI systems, this meta-methodology provides a flexible, adaptable approach that can be tailored to different domains and cultural contexts. The methodology is based on a Question–Answering mechanism, which guides the stakeholder through a structured flow of questions and answers, automating – behind the scenes – technical steps to build eventually a fair AI system. By leveraging a questionnaire, the system gathers contextual and domain-specific information, applying related socio-legal constraints to ensure fairness. This form of interaction allows making well-informed decisions, even without deep technical knowledge, consequently increasing the fairness problem awareness. The proposed approach is easily adaptable and evolvable, in order to keep up with the changes in the domain of the system under design, and to refine the methodology over time.

Mattia Matteini: *todo*

Contents

List of Figures

List of Listings

Chapter 1

Introduction

Fairness is a fundamental ethical and social principle that ensures impartiality, equity, and justice in decision-making processes. In the context of Artificial Intelligence (AI), fairness pertains to the development of software systems that do not demonstrate biases or systematic discriminations against specific individuals or groups. Achieving fairness in AI is a complex challenge, as many AI systems reflect historical societal biases and discriminations present in the data they are trained on. Moreover, the multidisciplinary nature of fairness requires integrating insights from computer science, ethics, law, and social sciences, making this challenge further complicated. Addressing these issues is essential to prevent damaging consequences in high-stakes domains such as hiring, criminal justice, healthcare, and finance.

GC: serve una citazione per ogni esempio

The implications of unfair AI are pervasive and far-reaching. Discriminatory algorithms can reinforce existing societal inequalities, disproportionately impacting marginalized communities. For instance, biased AI models in dermatology have led to incorrect diagnoses for skin diseases in black individuals due to training data lacking diverse skin tones [DVN+22]. Similarly, AI-driven loan approval systems have been documented to deny home loans to black people based on historical biases embedded in financial datasets [WYBM20]. Despite these examples, many people are not even aware of the fairness problem and consequences that can arise. For these reasons, lots of effort in research has been put in the last years to address this critical field.

Since AI capabilities and use cases have significantly increased in short time, there is a lack of engineering methodologies to develop fair AI systems. The objective of this thesis is to provide a (meta-)methodology to build fair AI systems. The methodology should assist both stakeholders and engineers in addressing fairness during AI systems creation. It has no ambition to be a one-size-fits-all solution, but rather a flexible and adaptable framework that can be tailored to different contexts and applications. The complementary objective is to reify the methodology into a software system, creating a practical tool easy to use and accessible also to non-technical people. The core of the methodology is a Question–Answering (Q/A) mechanism, that guides stakeholders through a structured flow of questions and answers. During this flow, the software system collects information about the system domain, rationalizes fairness-related issues, and automates technical steps in order to train and build—in background—a fair AI system. This work acts on the whole AI lifecycle, taking into account fairness considerations since the beginning of the process, reducing the presence of biases in the system. The methodology is designed to facilitate the translation of socio-legal requirements into technical constraints, because this represents one of the main challenges in the field.

It is worth highlighting that the contribution of this thesis is a central outcome of the Horizon Europe project "Assessment and Engineering of eQuitable, Unbiased, Impartial and Trustworthy Ai Systems" (AEQUITAS, G.A. 101070363)[1], which aims to promote fairness, accountability, and transparency in AI-driven systems, by involving a consortium of academic and industrial partners from different countries. Furthermore, a preliminary version of the work presented in this thesis has been accepted for publication in the Proceedings of the 58th *Hawaii International Conference on System Sciences* (HICSS) [CMS+25]. For all these reasons, our contributions have undergone a thorough validation process, involving not only the anonymous reviewers of the conference, but also the AEQUITAS consortium partners, which validated our work from both a conceptual and technical perspective. The practical software tool has been tested and validated through a series of focus groups involving experts and unprivileged groups. In addition, the methodology and its software implementation have been validated through project reviews, receiving positive feedback from external reviewers. Feedbacks have been

[1] https://cordis.europa.eu/project/id/101070363

used to improve and evolve the methodology and the corresponding to technology to their present form, and will contribute to their future development.

Structure of the Thesis. Chapter 1 introduces the topic of fairness in AI and outlines the objectives of the thesis. Chapter 2 provides a comprehensive overview of fairness, its significance nowadays, and the challenges associated with achieving it in AI systems. Chapter 3 presents the conceptual contribution to this thesis: a meta-methodology aiming to address fairness in AI systems. Chapter 4 illustrates the design of the produced software artifact that implements the meta-methodology. Chapter 5 goes deeper into the software implementation details, providing low-level and technical choices. Chapter 6 discusses the validation of the meta-methodology, highlighting strengths and limitations. Chapter 7 concludes the thesis, summarizing the core of the contribution and describing future research directions.

Chapter 2

Background

2.1 What is Fairness?

Fairness, from an ethical and social perspective, is the principle of treating individuals and groups equitably, ensuring that no one is unjustly advantaged or disadvantaged due to biases, discrimination, or arbitrary distinctions. It is deeply rooted in moral philosophy, legal systems, and societal norms, aiming to promote justice, equality, and inclusion. A just society requires reducing social inequalities, and ensuring that opportunities and resources are distributed in a way that acknowledges both individual merit and systemic disadvantages. The concept of fairness evolves based on cultural, historical, and contextual factors, reflecting a society's commitment to ethical treatment and social cohesion.

Fairness in AI. From a technical point of view, fairness in AI refers to the development and deployment of AI systems that minimize biases and prevent discriminatory outcomes. It involves designing systems that ensure equitable treatment across different demographic groups, particularly those historically marginalized or disadvantaged [JMCB22]. The main challenges in this field are building *fair-by-design* systems—in which fairness is addressed since the very beginning of the process—and detecting biases in already existing systems, mitigating them if possible [KBP+24].

Before the advent of fairness, AI systems were developed with the primary

goal of optimizing performance metrics, such as accuracy. Nowadays, that fairness is becoming a crucial aspect to consider, accuracy is no more the only metric to optimize. It is necessary to find a balance between accuracy and fairness. Besides that, fairness can also be in contrast with the performance of the model, making difficult to find a good trade-off between these two aspects.

Fairness is becoming crucial because AI systems increasingly influence decision-making processes in various sectors of society, including hiring [RFV24], lending [EPL23], and healthcare [UKF+24]. If AI models are biased, they can perpetuate and even amplify existing societal inequalities, leading to unjust outcomes and dangerous effects on individuals and communities [Fer24]. Ensuring fairness in AI enhances trust, transparency, and accountability, making AI systems more ethical, reliable, and beneficial for society.

AI has undergone significant advancements over the past few decades, causing an enormous increase in its adoption across various domains, until becoming pervasive in the daily life of people [Fer24]. This also caused a growing of biases in AI systems, as discriminations are intrinsically part of the human history, and consequently of the data that AI systems are trained on [Fer24].

AI is now widely used in critical domains, as previously mentioned, where biased decisions can have life-altering consequences [Fer24]. For instance, in healthcare, biased algorithms may lead to misdiagnosis or unequal treatment recommendations for different demographic groups; in finance, AI-driven credit scoring models can reinforce discriminatory lending practices, limiting access to financial resources [EPL23]; in the criminal justice system, biased predictive policing and risk assessment tools can disproportionately target marginalized communities [Jos24]. Given these risks, ensuring fairness in AI is essential to preventing discrimination, maintaining ethical standards, and safeguarding people.

On Multidisciplinarity. Achieving fairness in AI requires a multidisciplinary approach that integrates insights from computer science, ethics, law and social sciences. Technical methods alone cannot fully address fairness, as it is deeply tied to societal values, human rights, and legal frameworks. Socio-legal experts help define fairness principles, ensure compliance with anti-discrimination laws and analyze the societal impact. The intersection of these fields highlights that AI

fairness is not merely a technical challenge but a complex, multidimensional issue requiring collective effort and interdisciplinary research and collaboration.

An impactful example regarding the work of legal experts in the field of Artificial Intelligence is the *AI Act* [Mad21]. The AI Act, proposed by the European Union, is a comprehensive regulatory framework designed to ensure that AI systems are safe, transparent, and aligned with fundamental rights. It categorizes AI applications into different risk levels—unacceptable risk, high risk, limited risk, and minimal risk—imposing stricter requirements on higher-risk systems, such as those used in hiring, law enforcement, and healthcare. These requirements include transparency, human oversight, and bias mitigation. However, translating these legal constraints into practical technical steps is not trivial.

Concepts like fairness, accountability, and explainability are difficult to quantify, and AI models often operate as black boxes, making compliance complex. While the AI Act sets an important precedent for AI governance, its effective implementation requires further collaboration between policymakers, legal experts, and computer scientist to bridge the gap between regulation and technical feasibility.

Measuring Fairness. At one point, in order to assess the fairness of an AI system, is important to have a way to "measure" how much the system is fair and in what terms. Remarking what said before, fairness is very context-dependent, and there is not a single method to measure it.

The need to cover multiple aspects of fairness has led to the introduction of various *fairness metrics*—statistical formulas that quantify fairness in different ways, each capturing a slightly different aspect of fairness. These fairness metrics, are a set of indexes that can be used to detect biases in AI systems, and they can be used indeed to evaluate the fairness of a model.

In the following, are listed two of the most common fairness metrics used in the literature [IML23]:

- *Statistical Parity Difference* (SPD) [DHP+11] measures the difference between the probability of the privileged and unprivileged classes receiving a favorable outcome. This measure should be equal to 0 to be fair.

Formally it is defined as $SPD = P(\hat{Y} = 1|A = a) - P(\hat{Y} = 1|A = b)$ where A is the sensitive attribute, $\hat{Y}$ is the predicted outcome, and a and b are the privileged and unprivileged groups, respectively.

- *Disparate Impact* (DI) [FFM+15] compares the proportion of individuals that receive a favorable outcome for two groups, a privileged group and an unprivileged group. This measure should be equal to 1 to be fair.

  Formally it is defined as $DI = P(\hat{Y} = 1|A = a)/P(\hat{Y} = 1|A = b)$ where A is the sensitive attribute, $\hat{Y}$ is the predicted outcome, and a and b are the privileged and unprivileged groups, respectively.

2.2 AI Lifecycle

Since the very beginning of the AI era, the standard lifecycle consists of the following "traditional" steps: (i) data collection and processing, (ii) model training, (iii) system evaluation. Obviously, this workflow in the latest years have increased in complexity and now, with the newer innovations and powerful models and architectures, it may appear even almost minimalistic, but it still represents the core of all AI systems. However, when fairness is taken into account, each step needs to be revisited in order to obtain an equitable, impartial, and fair AI system.

To achieve this goal, the technical perspective is not enough. Fairness is a multidisciplinary concept that involves social, legal, and ethical aspects. Therefore, the AI lifecycle needs to be constrained by socio-legal requirements that engineers must consider during the development process. This includes understanding the societal impact of AI systems, ensuring compliance with legal standards, and adhering to ethical guidelines.

There are also many differences between the socio-legal and technical perspectives. Regarding the AI lifecycle, engineers tend to focus on technical aspects and few development phases, in fact the major part of the literature speaks only about *pre-processing*, *in-processing* and *post-processing* (Figure 2.1). Respectively, *pre-processing* involves data collection and preparation, *in-processing* refers to the model training phase, and *post-processing* deals with the fair evaluation of the AI system.

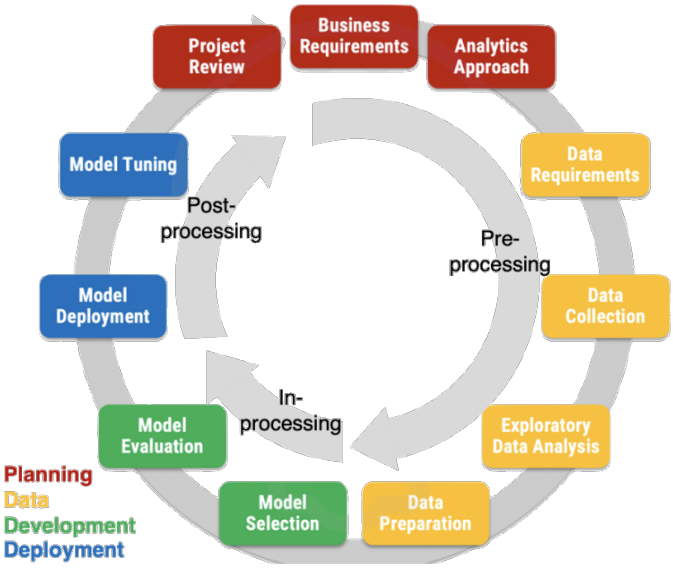


Figure 2.1: Fair AI lifecycle from [CCMO23]

Often engineers adopt reductionist approaches addressing a field that is not their own, discarding the big picture of social, economic, and institutional constraints. On the other hand, socio-legal experts consider a broader range of activities and phases. They focus on "building blocks" for fair AI such as risk assessment, stakeholder identification, regulatory analysis, and fundamental human rights impact assessment. In particular, with respect to fundamental rights impact assessments, these will be legally required for some AI systems, yet no standard for implementing them has emerged so far.

2.3 Practical Issues

2.3.1 What is (un)fair?

Fairness in AI (and beyond) is inherently subjective, shaped by cultural values, ethical theories, and individual perspectives. What one group considers fair may not align with other people's understanding, leading to debates about determining what is just and what is not [JMCB22]. This subjectivity and variation in viewpoints complicates efforts to develop standardized fairness metrics, as no single approach can universally capture the diverse and often conflicting notions of

fairness present across different social, legal, and institutional contexts.

Beyond its subjectivity, fairness is also highly context-dependent. The same algorithm might be considered fair in one application but biased in another, depending on the societal, legal, and institutional constraints surrounding it. For instance, fairness considerations in hiring algorithms differ from those in criminal justice risk assessments, necessitating tailored approaches rather than generic solutions. Moving forward, privileged and unprivileged groups change depending on the application domain, as well as the fairness criteria that are taken into account.

2.3.2 Bridging Perspectives

Bridging the socio-legal and technical perspectives on fairness is a significant challenge. Guidelines and descriptive methodologies exist to address fairness compliance from a social-legal perspective, but their approach offer broad guidelines without defining practical steps, leaving interpretation to technical experts [CMS+25]. The lack of alignment between these viewpoints makes it difficult to translate abstract fairness principles into concrete computational methods. This also leads to a proliferation of metrics, each measuring slightly different aspects of fairness, reflecting the diverse priorities and domain perspectives.

A fundamental obstacle to this integration is the differing language used by socio-legal experts and technical people. It is difficult to reach an agreement if even a concept or term can assume different meanings depending on the perspective. This linguistic division creates a barrier to interdisciplinary collaboration, leading to misunderstandings even when working towards shared goals.

These perspectives are shaped also by distinct academic and methodological backgrounds. Legal and ethical frameworks tend to be verbose and highly context-specific, relying on various interpretations and case-by-case analyses. In contrast, technical disciplines prioritize concrete steps and pragmatic aspects.

In literature, there is a lack of methodologies regarding the building of fair AI systems. The lack is not just related to the technical perspective, but also to the socio-legal one. This is enhanced by the fact that design and develop a single methodology fitting all kinds of AI systems is not feasible, as the system requirements and constraints change depending on the context and the applica-

tion domain. Of course, the creation of such methodology is complicated by the multidisciplinary complexity of the problem, and should involve expertises across all the relevant fields.

Chapter 3

The Meta-Methodology

A **methodology** is a structured framework that outlines the principles, processes, techniques and best practices used to conduct research or develop systems in a systematic and reproducible manner. In this context, a well-defined methodology would be essential for ensuring fairness, as it would provide a rigorous approach to (i) translating socio-legal requirements into technical steps, (ii) identifying and mitigating biases, and possibly (iii) building *fair-by-design* systems.

Having a rigorous methodology would positively impact the development of fair AI systems. It would represent a clear path to follow, encapsulating the already existing unclear guidelines provided by the socio-legal frameworks. Unfortunately, factors such as multidisciplinarity, complexity, and context-dependency make it difficult to design a single methodology that fits all contexts and applications. Therefore, this contribution proposes a **meta-methodology** that provides a flexible and adaptable framework for design and develop multiple methodologies instead of a single one. The idea of such meta-methodology comes from the sessions of brainstorming and discussions between experts of different fields, where troubles have emerged in reaching an agreement and proceeding with clear technical steps relying on the legal requirements.

3.1 Desiderata

In the following, are listed the requirements that the meta-methodology should satisfy.

R1 Requirements Translation: The methodology should assist experts in translating the socio-legal requirements into practice.

A big challenge in this field, is to understand how legal constraints can be applied, and how technical steps can adhere to the requirements. That's why the methodology should provide a mechanism assisting this phase.

R2 Context and Domain Awareness: The methodology should consider the cultural context and the domain of AI system under design.

AI systems have been applied in several (and critical) use cases. For each of them, the constraints and requirements change, hence, through the methodology, it should be possible to understand the system domain and be context-aware.

R3 Adaptability: The methodology should adapt to any change in the cultural context as it evolves.

Some context could be volatile in terms of societal norms and cultural changes, so the methodology should be able to adjust and align to new constraints.

R4 Building the AI System: The methodology should not just provide a theoretical guideline, but also assist the AI system creation.

This means that it is necessary a software reification of the methodology permitting to be applied practically, obtaining eventually, a fair AI system.

3.2 Concepts

The Roles. In the proposed methodology process, there are two main roles involved:

1. **Business User (BU)**, also called **stakeholder**, who is the person commissioning the AI system.

GC: forse è meglio usare stakeholder come sinonimo di BU: nel paper intendevamo un'altra cosa, come puoi evincere anche dalla figura 3.1

2. **Technical User (TU)**, who is the person with technical background, assisting the BU in the development of the AI system.

With respect to BU, it is assumed that he or she may have limited or no technical knowledge. This is a common scenario in the real world, where often stakeholders are people with a specific domain expertise, but not necessarily with technical skills. One of the goal of the methodology, is to provide a way to assist the BU in the development of the AI system, without requiring necessarily deep technical knowledge. Potentially, stakeholder could even build a fair AI system without the need of a TU.

On the other hand, the Technical User, despite is the person with technical background, is not the responsible for the entire system development. Firstly, TU must be able to assist BU during the process to clarify any technicalities that may arise, and secondly, they must have some knowledge about statistics, and AI fairness.

Finally, TU may contribute to the system development through the implementation of scripts/computational processes involved in the building of the system and integrated in the methodology. In fact, the software reification of the methodology will be a tool providing APIs for technical people, in order to permit them to attach their scripts.

Questionnaire. Discussions among experts from involved fields highlighted the need for a practical understanding of the domain of the system being designed, ensuring that it is accessible and comprehensible to people of any background. The proposed approach relies on a straightforward questionnaire, which directly engages the Business User with questions regarding the system's domain.

The questionnaire serves as a structured flow of questions and answers designed to gather essential contextual information. Depending on that, it provides practical steps to guide the development of a fair AI system. Questionnaire is not just used to collect information, it also acts as a tool to assist the BU in making well-informed decisions. At the same time, questions represent also technical steps to be taken, addressing the socio-legal constraints in a comprehensible way. This approach is central to the methodology, owing to its simplicity and versatility in

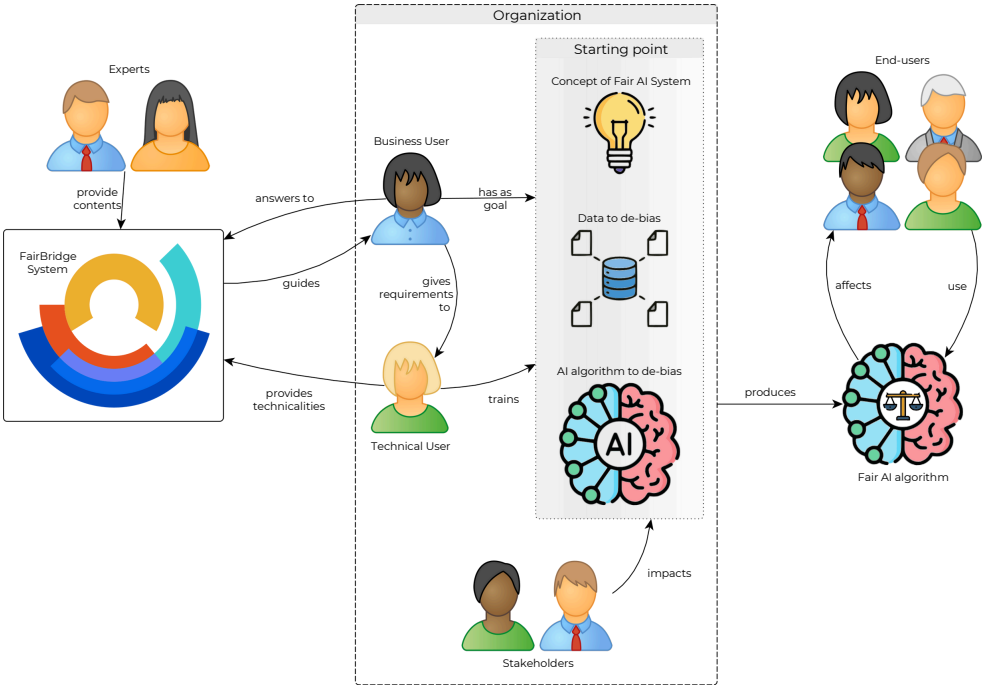


Figure 3.1: Concept of the proposed approach to fairness engineering from [CMS+25].

capturing constraints and supporting multiple use cases. The overall concept is shown in fig. 3.1.

The pool of questions and answers should be designed ad-hoc from a team of multidisciplinary experts. This is a crucial, and non-trivial, step in the methodology, as the questions should be able to capture the constraints and requirements from legal frameworks, but also to provide technicalities to be addressed in the proper way and at the right time in the process.

Examples of questions that could be asked are:

- "In what area will the AI system be applied?" (Healthcare, Finance, Hiring, etc.)
- "Do you have some AI system already in place, or are you developing an AI system?"
- "Is the dataset sufficiently representative of the population where the system will be used?"

But also more technical questions like:

- "What are the fairness metrics that should be considered?" (Statistical Parity Difference, Disparate Impact, etc.)
- "Which are the proxies for the sensitive features?"
- "Which data mitigation algorithm do you want to use?" (Disparate Impact Remover, Learned Fair Representations, etc.)

Order of Questions. As mentioned in the previous paragraph, the flow of questions comprises both generic and technical questions. The questionnaire should follow a structured approach, beginning with general questions before and gradually introducing more technical aspects. Initially, broad and non-technical questions are asked to establish a clear understanding of system's domain, purpose, and the cultural or business context in which it operates. As the questionnaire progresses, questions become more specific and technical. At one point it becomes mandatory to introduce technical aspects because, in the end, questionnaire has to converge to the effective building of the fair AI system.

There is another important concept related to the order of questions: the answer to a question can **influence** the following ones. This feature is to enable the methodology to adapt to the context and asking later more tailored questions based on the previous answers. Moreover, it is also useful to enrich the part of system development, as it should be possible to follow multiple paths to make an AI system fair. For instance, the Business User could decide to preprocess the dataset, or choose to perform just in-processing mitigation. These mechanisms lead to a more flexible and adaptable questionnaire, capable of addressing a wide range of contexts and applications, enabling also branching and joining paths in the flow.

Decision Support. The methodology includes—alongside the Q/A—a Decision Support Mechanism, aiming to simplify the process of making decisions regarding fairness-related or complex questions. Fairness problem should be taken into account not just by experts in the field, but also by stakeholders, as this problem is

becoming more and more important. Therefore, an important goal of the methodology is to make the BU aware of the fairness problem, and to assist him/her in making well-informed decisions. In this way, BU can gain a deeper understanding of the topic, and can proceed with the development of the system more responsibly. Importantly, the mechanism does not impose decisions but rather **suggests** to the BU the answer he or she probably should give.

In addition to showing questions, any software reification of our methodology should provide supplementary information and resources, like charts and tables, helping the BU to gain a better insight of what he is doing and to assess the fairness of the system more effectively.

3.3 The Q/A Mechanism

The Question–Answering (Q/A) mechanism represents the core of this methodology. It has been the starting point to bridge the gap between socio-legal and technical perspectives and provides a structured way to "translate" the legal constraints into technical steps, contextualizing them in the application domain. Behind the scenes, the Q/A mechanism is a **directed graph** that represents the decision-making process.

About the Graph. Formally, the graph is defined as $G = (V, E)$, where V is the set of nodes and E is the set of directed edges. **Nodes** consist of two distinct types: *question nodes* and *answer nodes*. Each question node contains a natural language sentence expressing an inquiry, plus an identifier (unique within the whole graph) such as `Q1`, `Q2`, etc. They can also contain other arbitrary information, like the type of question (single or multiple choice), a brief description, and so on. Answer nodes, similarly, contain a natural language sentence expressing a possible answer to a question, an identifier (unique within the whole graph) such as `Q1-A1`, `Q1-A2`, etc., and other arbitrary useful information. **Edges** are of two sorts too: either *question-to-answer* edges, denoted by $Q \to A$, or *answer-to-question* edges, denoted by $A \to Q$. Edges of the first type ($Q \to A$) indicate that question Q has as a possible answer A, whereas edges of the second type ($A \to Q$) represent that next question to be asked is Q if the selected answer to the previous question was

A.

These statements assume that there cannot exist links between two question nodes or two answer nodes. It is also assumed that each question must have *at least one* outgoing edge leading to an answer node, and each answer node must have *exactly one* outgoing edge towards a question node. Moreover, at the current state, if a question is of multiple choice, it is assumed that each possible answer leads to the same next question.

Proceeding with the graph details, the **root** node is the first question to be asked to each BU, while **leaves** represent the answers to the last question, which are, technically speaking, answer nodes with *no outgoing* edges.

Finally, it is assumed that the graph is *connected*: each questionnaire is guaranteed to have a beginning and an end, meaning that there is *at least one* path from the root to each leaf.

Why a Graph? This structure ensures that the graph alternates between questions and answers, forming a coherent flow of a typical Q/A session. Moreover, this type of graph is particularly suitable for representing decision-making processes, as it allows for a structured flow of questions and answers, guiding the user through a series of steps. The graph may contain cycles, allowing the repetition of some questions (and steps), giving even more flexibility to the process.

With the branching feature, it is possible to follow multiple paths, depending on the answers given by the Business User (BU) (and so depending on the context). Remarking the order of questions, the graph effectively encodes a deterministic, yet non-sequential, flow of questions and answers, where each answer directly influences the next question to be asked.

The graph is a data structure that can evolve and change easily over time, as the methodology actually is a **meta-**methodology, it is possible to create multiple versions of the graph. For instance, it is possible to design different pools of questions and answers, and indeed different graphs, each one tailored to a specific context or application domain. Furthermore, this flexibility enables also the possibility to adapt to any change in the cultural context taken into account, leading to a possible "methodology versioning". In fact, changing the graph structure means creating a "new version" of the methodology. New version denotes a graph with

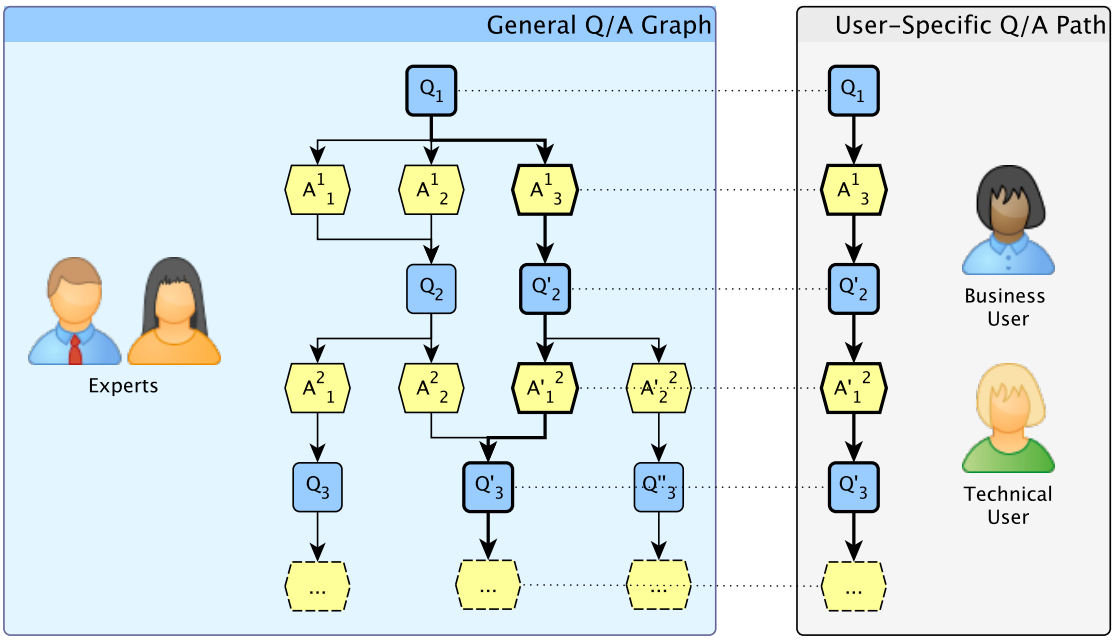


Figure 3.2: A graphical representation of the Q/A mechanism, viewed as a graph by experts and as a sequential path by business and technical users (from [CMS+25]).

different paths of questions and answers, which obviously, addresses diverse needs. Hence, the methodology can adapt to evolvable cultural contexts, creating new graphs for each needed change in the application domain.

General Graph vs Project-Related Graph. So far, it has been described the general blueprint of the Q/A: how to represent questions and answers, how the graph is structured, how the questionnaire and all its features look like. However, from the Business User (and Technical User) perspective, the sequence of questions seems a linear path. It is intentioned that, the person involved in the process, has the feeling of just compiling a questionnaire, where all the complexity is kept out of sight. In fig. 3.2 is shown a graphical representation of the Q/A mechanism, where the flow of questions from users' perspective is defined by a path in the graph.

The end user can navigate freely through the questionnaire, answering questions and returning to previous ones in case of need. Of course in this way it is possible to change the path, and so the flow of future questions.

3.4 Automation

At this stage, it is yet vague how the proposed methodology can inject fairness measures and technical steps into the development process. Here is the point where Technical User (TU) contributes to the process, by implementing scripts and computational operations that are eventually integrated into the final system. Said that, TUs play an important role because, regardless of the complexity and variety of the needed scripts, such scripts can change case by case. Moreover, TU is still useful as source of technical knowledge and assistance for the Business User (BU).

GC: non capisco perchè ogni tanto riespandi questi acronimi

However, while many activities performed by technical users are specific to their respective organizations, certain tasks are generalizable enough to be automated directly by the implemented methodology. Relevant examples of this, are the computation of fairness metrics and the identification of biases in datasets. These, in fact, are enough consolidated to be automated and integrated into the methodology. Rather than requiring individual technical users to develop their own solutions from scratch, the system itself can integrate these capabilities as built-in system-level functions, ensuring consistency, efficiency, and reliability across different projects.

The reification of the meta-methodology should be purposefully designed to facilitate this progression toward greater automation. When certain actions——such as evaluating responses or detecting biases——are widely applicable rather than organization-specific, they can be implemented as reusable system-level solutions. This eliminates redundancy, reducing the need for technical users to repeatedly address the same challenges independently.

These scripts, designed to be injected into the methodology, can be provided by the software implementation of the methodology itself, developed by technical users within organizations, or contributed by third-party developers. This flexible approach ensures that automation capabilities can expand over time, adapting to emerging best practices. In the early stages of system adoption, technical users may handle certain tasks manually, but as the methodology matures, these tasks can gradually be automated.

Technically, these scripts can be attached easily to the software implementation

of such methodology, and they can be triggered whenever some kind of events occur, like the answer to a question, or termination of another computation.

Chapter 4

Software Design

The goal of this chapter is to design a software artifact that reifies the meta-methodology proposed in Chapter 3, in order to make it usable by stakeholders to develop fair AI systems. Software design must be flexible and adaptable, as the methodology is intended to be subject to changes and improvements over time.

The final product is designed to be used by Business Users (BUs) through an intuitive web interface. Hence, frontend will interact with the underlying backend which handles the Q/A mechanism and automates technical steps.

4.1 Architecture

The software adopts an **event-driven** architecture (fig. 4.1), as events are used to handle part of the communications between the components. This design choice is motivated by the need to trigger automation scripts at specific points, for instance, when a question is answered or when a computation is completed.

The entire system is composed of the following components:

1. **Backend**: the core of the system, managing the Q/A mechanism and the automation of technical steps.

2. **Database**: component handling persistency of questions, answers, and other relevant information.

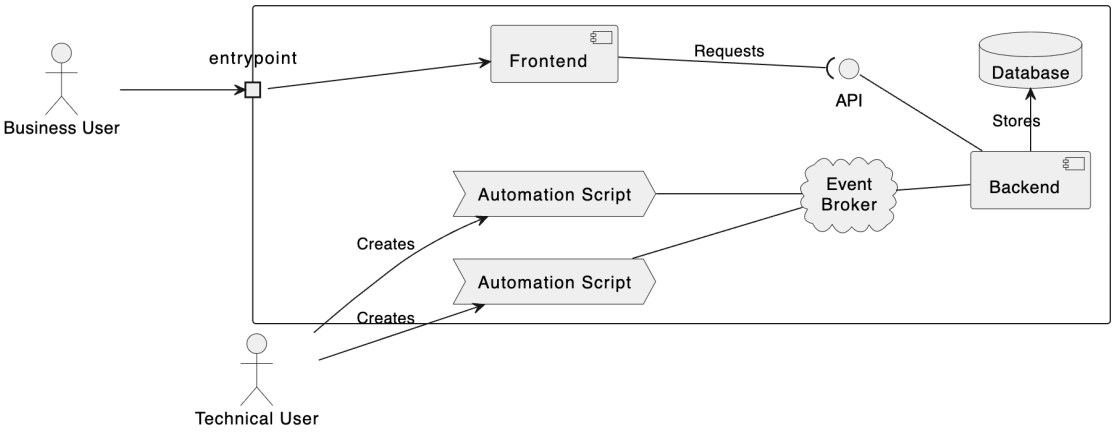


Figure 4.1: The overall software architecture of the proposed software system.

3. **Automation Scripts**: a set of pluggable scripts that automate technical steps, such as computing fairness metrics and mitigating biases.

4. **Event Broker**: component responsible for event handling in the whole system.

5. **Frontend**: the web application that allows BUs to interact with the system.

Clean Architecture. Backend component is the core of the system, it is a web service encapsulating the business logic managing the Q/A mechanism. It has been designed using Clean Architecture [Mar17], in order to separate the business logic from technical details, making the system flexible and technology-agnostic. This means that high-level business logics do not depend on low-level implementation details. And besides, it improves the separation of concerns, separating clearly core concepts, business rules, and technical details.

Therefore, backend is divided into the following layers:

1. **Domain**: the layer containing the core domain entities.

2. **Application**: the application-specific business logic.

3. **Presentation**: the interface between domain entities and the external technologies.

4. **Infrastructure**: the outermost layer, containing the implementation details technology-specific.

The dependency flow is unidirectional, from the outermost layer (Infrastructure) to the innermost one (Domain). So, for instance, Application layer and Presentation layer can depend on Domain layer, but not vice versa.

4.2 Structure

The considered design choices follow the principles of **Domain Driven Design (DDD)** [MT15], a software design philosophy that emphasizes the importance of domain model in the development process. The main goal of DDD is to align the software system with the domain model, ensuring that the software reflects the real-world domain as closely as possible. Although the focus of this contribution is not on the software design phase in itself, the adoption of DDD is motivated by the volatile nature of the methodology, which is expected to evolve through contributions and improvements from multiple people.

Projects. In order to introduce the main Q/A entities, it is necessary to define the concept of `Project`, which is the association with the AI system that the BU wants to build. In fact, each `ProjectQuestion` is related to (and also identified by) a specific `Project`. This is not just a way to distinguish multiple AI systems creations, but also to encode and store project-specific information in a "store" called `ProjectContext` (or only `Context`). This store is essential because each AI system has its own dataset, features, algorithm, and so on. Since these data can be of any nature, and are strongly volatile, the idea is to set up a key-value store, without any kind of constraints, where arbitrary (encoded) information can be stored. For instance, at some point it will be useful to store a key-value pair where the key is the name of the dataset, and the value is the encoded dataset itself. These data then will be available and useful to automation scripts and backend business logic.

Q/A mechanism. Q/A mechanism can be intuitively mapped to two main **domain entities**: `Question` and `Answer`. However, it is necessary to distinguish

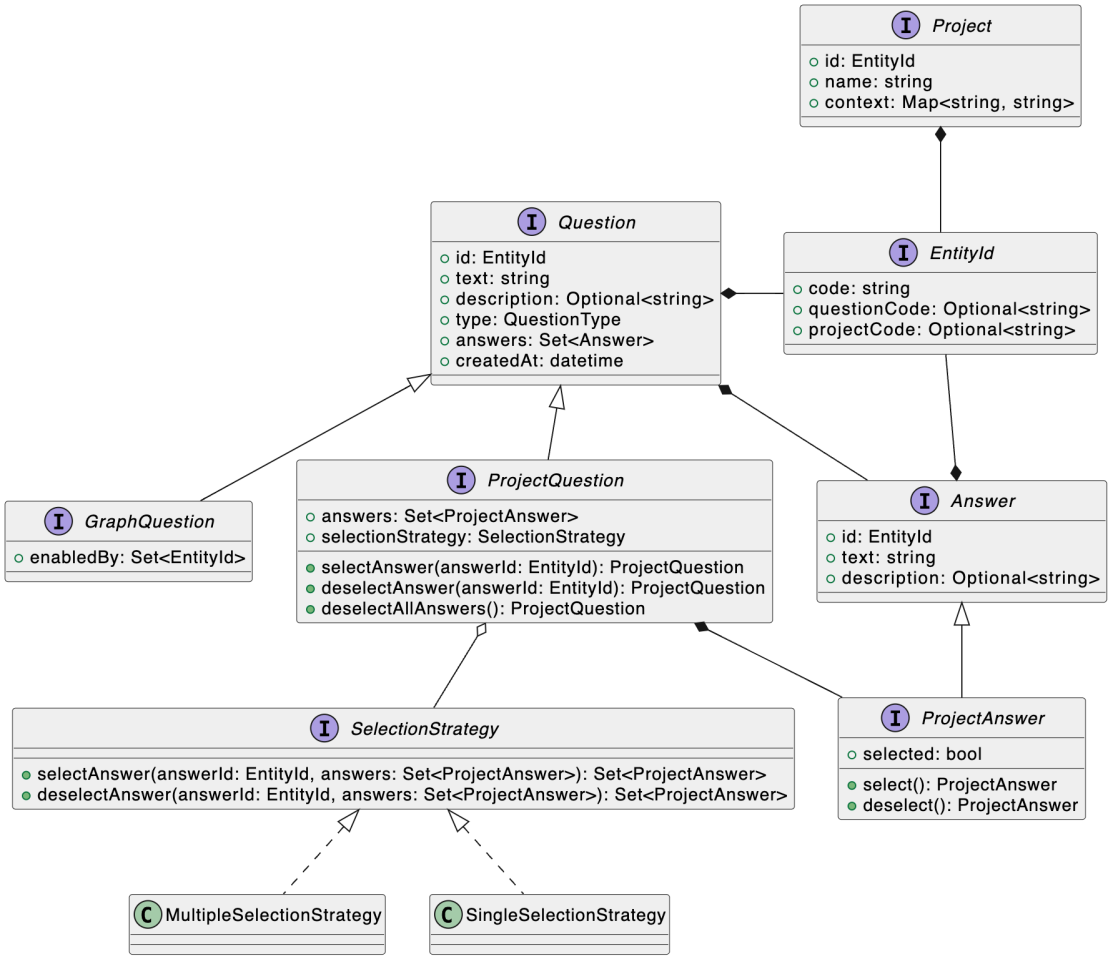


Figure 4.2: UML class diagram of main Q/A entities.

between questions and answers that are part of the general graph and those that are project-related. Therefore, in the domain model are present the related entities as shown in fig. 4.2.

`GraphQuestion` represents an extension of the general `Question` entity, containing a set of answer ids that are meant to "enable" such question in the graph. In other words, a `GraphQuestion` is a question that is part of the general graph, and the set of answer ids is needed to encode the *answer-to-question* edge ($A \to Q$) (see section 3.3). With regard to admissible answers, for the `GraphQuestion` is sufficient to rely on the general `Answer` entity because it doesn't need to have more information.

On the other hand, `ProjectQuestion`s are part of the project-related graph, and each of them is related to (and also identified by) a specific `Project`. The main difference between `GraphQuestion` and `ProjectQuestion` is that the latter has the possibility to "select" a certain answer (in the case of single choice questions) or multiple answers (in the case of multiple choice questions). Indeed, in this case, answers must have a state representing whether they are selected or not. `ProjectAnswer` entity, in fact, is intended to fulfill this need, as it contains a boolean field, and a `SelectionStrategy` handling the logics behind the selection.

For each **domain entity**, other DDD building blocks are defined:

- **Factories**: to facilitate the creation of new instances.
- **Repositories**: to manage the persistence of the entities.
- **Services**: to handle the business logic related to the entities.

From the structural view point, factories and repositories reside in the Domain layer, as they are intended to simplify the management of domain entities. Services instead, are part of the Application layer, since they compose the entire business logic of the system. However, the persistence of the entities strongly depends on technological details, so, while contracts are defined in the Domain layer, the actual implementations are pushed away to the Infrastructure layer (as shown in fig. 4.3).

The same applies to the `EventsService`, which is a particular service handling the event production and consumption. The reason is that an event-driven architecture can be achieved using different technological solutions, and business logic should remain technology-independent. Furthermore, due to this layer organization, it is possible to easily interchange the underlying technologies without affecting business logic and core domain entities (for instance using a relational database instead of a NoSQL one).

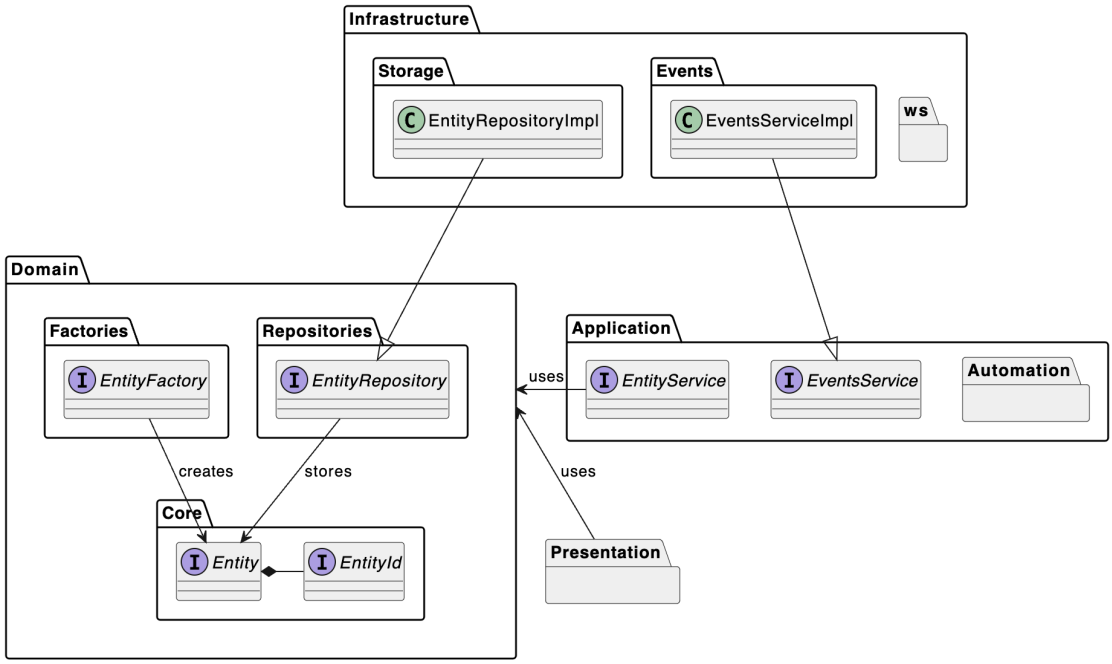


Figure 4.3: Representation of packages structure.

4.3 Behavior

The system behavior is mainly concentrated in **Backend** and **Automation Scripts** components.

Mattia Matteini: backend forse brutto nome? meglio Q/A service?

4.3.1 Backend

Backend component is a web service that exposes a REST API, enabling clients to interact with questions graph and questionnaires. Its primary functionalities include modifying the questions graph structure (adding, removing, modifying questions) and managing project-related questionnaires compilations. It is also possible to create new projects, and, for each of them, starting a new questionnaire session. Summarizing, backend is a simple listening web server, that manages requests and trigger events when relevant actions occur.

An important aspect to notice in this component behavior, is the `Context` update of various projects. `Context` updates are essential to the correct business logic functioning, because the major part of fairness-related computations strongly

depends on the specific data stored in the `Context`. For example, all data mitigation algorithms require the actual dataset, or the computation of fairness metrics needs the dataset and the selected output and sensitive features. Such computations, can be executed by both backend and automation scripts, for this reason `Context` updates represent a key point in the system behavior.

4.3.2 Automation Scripts

Automation Scripts are responsible for responding to specific domain events, performing necessary (fairness-related) computations, and updating system state. These scripts act as event-driven processes that listen for predefined triggers, such as selecting an answer to a question, creation of a new dataset, or completion of a fairness metric computation. Once the processing is completed, they send relevant updates back to the backend via API requests, ensuring that the system remains up-to-date and operates seamlessly without requiring manual intervention.

4.4 Interaction

The starting point of interactions is BU, engaging with the system through the Frontend component. Frontend is in charge of presenting questions to the BU, collecting answers, and sending them to the backend service. Backend, in turn, processes the requests and updates the questionnaire state accordingly. At this point, backend service triggers an event for each relevant action undertaken, which is caught by all Automation Scripts previously subscribed to the Broker. Each script performs its computation, and eventually sends back the results to the backend, which updates the system state and notifies the frontend to display the results (if there are any) to the BU. It is also possible that, during computations, are triggered other events for other relevant actions, like the processing of a new dataset.

The entire interaction flow is shown in fig. 4.4.

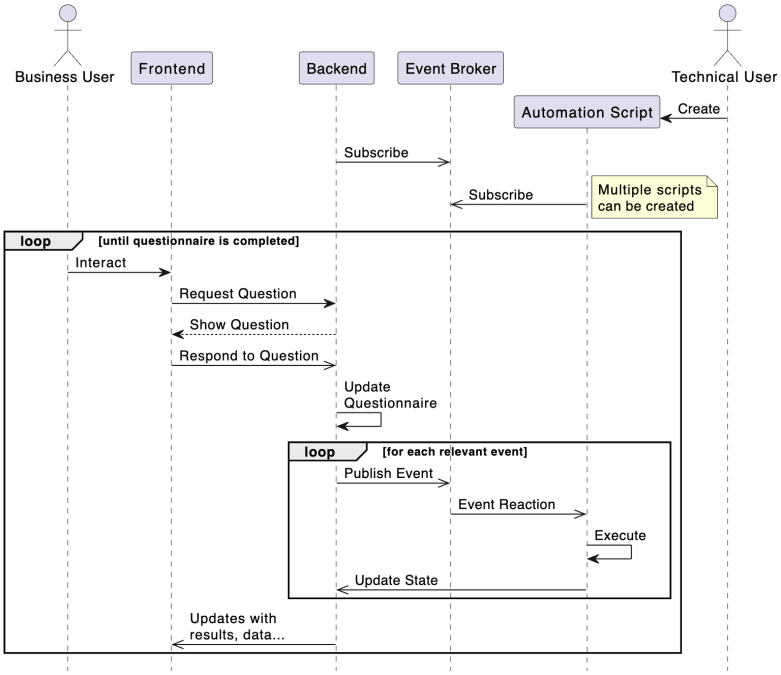


Figure 4.4: Generic sequence diagram showing the interactions between main components.

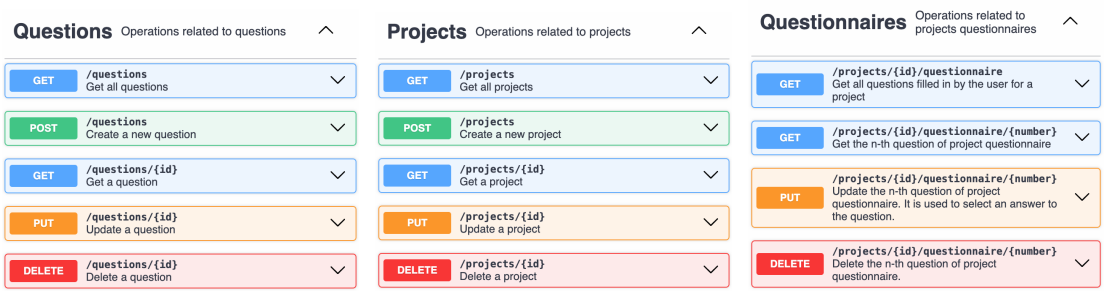
4.5 API

4.5.1 Rest

Backend service exposes an API used to interact with the three main entities: **projects**, **questions** graph, and **questionnaires**. Resources are organized following the REST principles, and each resource is identified by a unique URI. Since questionnaires are project-related, such resource is under the `Project` hierarchy. In fig. 4.5 are shown the main endpoints of the API.

4.5.2 Events

The other method available to interact with the system, which powers also internal communications, is the event-driven one. This represents an API slightly different from the classic REST one. Here, API endpoints are channels on which operations of type `send` and `receive` are performed. This type of API enables asynchronous communication between system components, in this case, the backend and au-



(a) API for `Question` operations (b) API for `Project` operations (c) API for `Questionnaire` operations

Figure 4.5: Backend REST API endpoints.

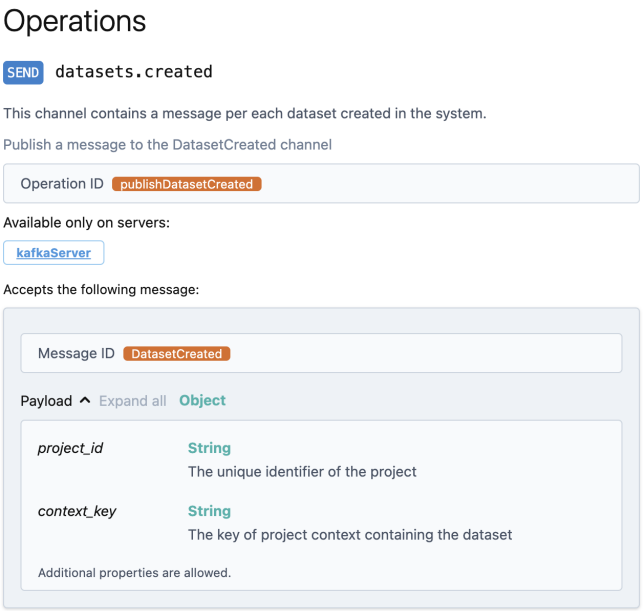
tomation scripts. That's fundamental because computations are intended to be non-blocking in order to allow BU to proceed in the questionnaire compilation.

A documentation example of such API is shown in fig. 4.6 using Async API specification[1]. The main concepts in this are:

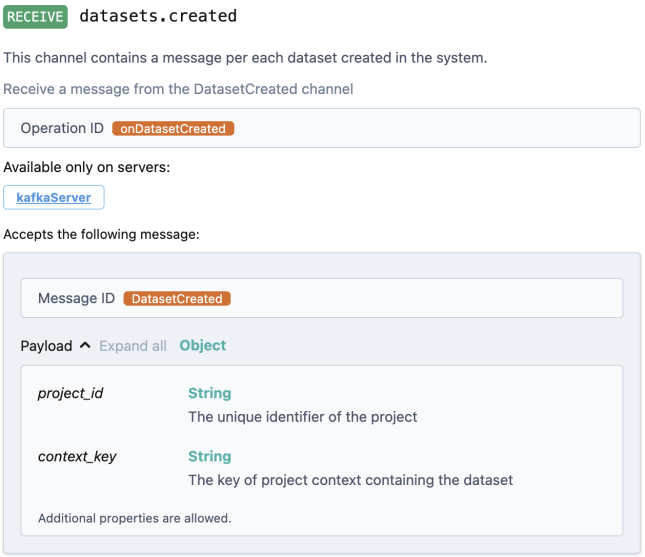
- **Channels**: Specific topics where events are published. Each topic corresponds to a particular type of event.
- **Operations**: Primarily `send` (`publish`), to publish events on the channel, and `receive` (performing if `subscribed` to channel), to receive messages when events are published.
- **Messages**: Schemas of the objects published on channels.

This API is highly flexible, in fact, when an automation script is created, it can be attached to a new channel created ad hoc for the script specific purpose. For instance, a new script can be plugged in simply creating a new channel and subscribing at it.

[1] `https://www.asyncapi.com`



(a) Publish operation for a channel



(b) Receive operation for a channel

Figure 4.6: Async API channels example.

Chapter 5

Implementation

This chapter delves into technical details of the software implementation which reifies the proposed meta-methodology. It provides an overview of the deployed architecture, and, basing on the design choices made in the previous chapter, it explains how the various components are actually implemented. The software implementation can be found on Github[1].

5.1 Components

Backend. Backend component is implemented as a web service in Python. The choice aims to reduce the abstraction gap because python comes with consolidated frameworks that facilitate development of AI systems. Python represents also a good choice thanks to its readability and simplicity, which make it easier to be maintained and extended by developers who were not originally involved in the project. Flask-restful[2] framework has been used to ease the development of the web service, as it allows the creation of a RESTful API with few lines of code (listing 5.1).

[1] `https://github.com/aequitas-aod/aequitas-backend` and `https://github.com/aequitas-aod/aequitas-frontend`

[2] `https://flask-restful.readthedocs.io/en/latest/`

Listing 5.1: Example resource creation with Flask-restful.

```python
questions_bp = Blueprint("questions", __name__)
api = Api(questions_bp)


class QuestionResource(Resource):

    def get(self, question_code=None):
        # ...

    def post(self):
        # ...

api.add_resource(
    QuestionResource,
    "/questions",
    "/questions/<string:question_code>"
)
```

Listing 5.2: Events Service implementation using Kafka in Infrastructure layer.

```python
from application.events import EventsService
from infrastructure.events import Producer, Consumer


class KafkaEventsService(EventsService):

    def __init__(self):
        self._producer = None

    def publish_message(self, topic: str, message: str):
        if self._producer is None:
            self._producer = Producer()
        self._producer.produce(topic, message)

    def start_consuming(self, topics: List[str], handler):
        Consumer(topics, handler).start_consuming()
```

Event Broker. Apache Kafka[3] is used as the event broker to implement the event-driven architecture. Kafka is a distributed streaming platform that provides high throughput, scalability, and fault tolerance. The python client of Kafka has been used to interact with the broker, allowing backend and automation scripts to publish and subscribe to events. This is achieved by an ad hoc service implemented in the infrastructure layer (listing 5.2).

[3] https://kafka.apache.org/

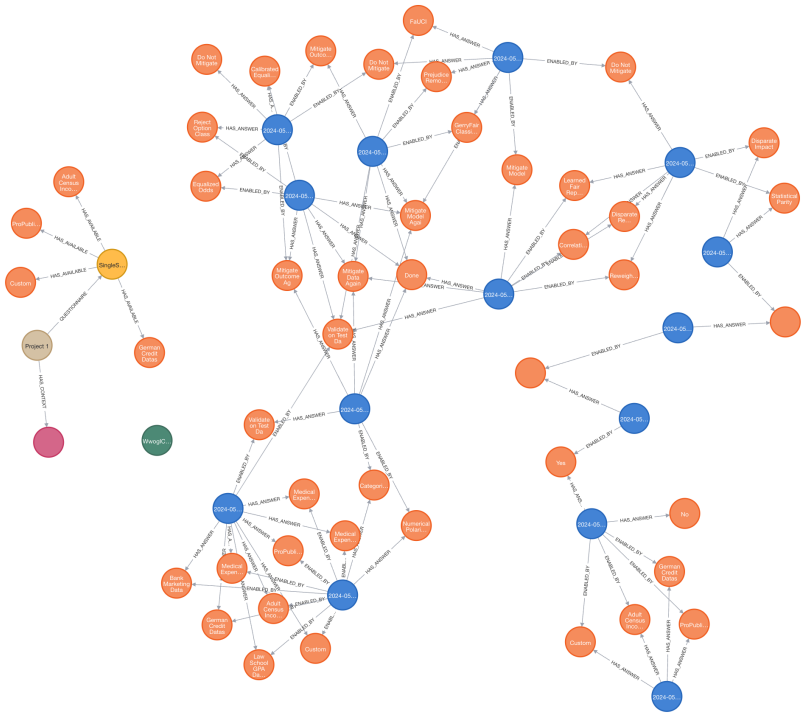


Figure 5.1: Visual representation of data in the graph database.

Database. For the persistency, the choice fell on a graph database, as it fits well with the graph structure of the Q/A mechanism (described in section 3.3). The technical implementation chosen is Neo4j[4]. It allows for efficient traversal and querying of the graph, making it easier to manage operations on questions and answers.

Entities are mapped to the database as nodes and relationships (see fig. 5.1). In particular, **questions graph** is mapped as the following: questions and answers are represented as nodes, while relationships between them are represented as edges. More precisely, let's consider Q_G is the node representation of question Q and A_G is the node representation of answer A (both in general graph). Edges in Neo4j are associated with a label, so edges of type $Q \rightarrow A$ are stored as $Q_G \xrightarrow{\text{HAS\_ANSWER}} A_G$, while edges of type $A \rightarrow Q$ are actually stored as $A_G \xrightarrow{\text{ENABLED\_BY}} Q_G$.

With respect to `Project`, let's consider P as its node representation. Key-value

[4] `https://neo4j.com/`

pairs of its `Context` are stored in P. This is straightforward because each node in Neo4j can store arbitrary key-value pairs, making it easy and flexible to save context data. In order to save space, context data that are common to all projects are stored in a separate node named `PublicContext`, which is referred if a key is not found in the specific project node on which the context data is requested.

Eventually, each **questionnaire** (if exists) is linked to a specific project P through a relationship $P \xrightarrow{\text{QUESTIONNAIRE}} Q_P$, where Q_P is the root of the questionnaire graph (the first question asked), and it is created upon the root of the general graph. Next questions in the questionnaire path are linked to the previous ones through relationships of kind $Q'_P \xrightarrow{\text{NEXT}} Q''_P$. To keep track of the answers given by the BU, each answer is linked to the corresponding question through a relationship $Q_P \xrightarrow{\text{HES\_SELECTED}} A_P$, where A_P is the node representation of the answer given by the BU.

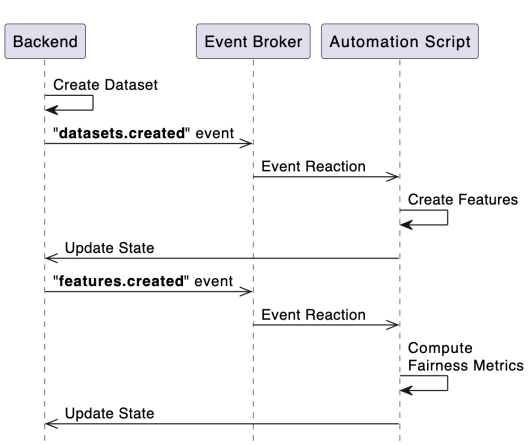
Automation Scripts. Automation scripts are processes created ad hoc to perform specific computations. They just need to adhere to the Async API specification to be seamlessly integrated into the system.

These scripts can actually be implemented in any language, but the ones created so far are written in Python. One motivation is that this allows scripts to share backend code, benefiting from reusability. Python is also recommended due to its extensive support for machine learning libraries and frameworks. At the moment, the main events identified are: (i) `questions.answered`, triggered when a question is answered; (ii) `datasets.created`, triggered when a new dataset is created; (iii) `features.created`, triggered when features of a dataset are analyzed; (iv) `processing.requested`, triggered when a processing request is made.

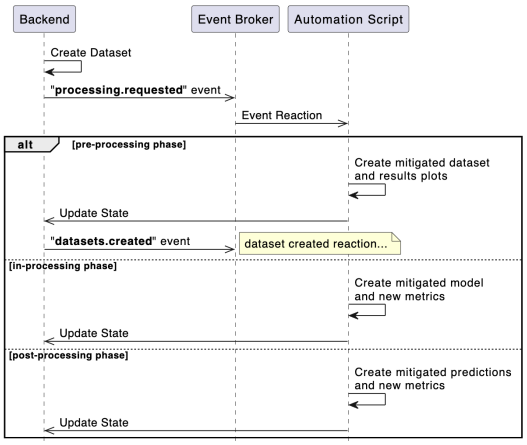
In fig. 5.2 are shown detailed sequence diagrams of these events.

Frontend. Frontend is implemented as a Single Page Application (SPA) using React[5] and Next.js[6]. This choice leverages the component-based architecture of React, which promotes reusability and maintainability of the code. Next.js enhances the development experience by providing server-side rendering and static

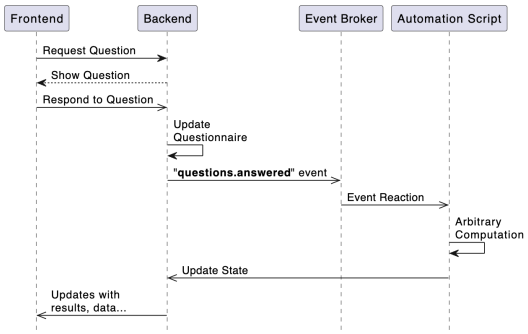
[5] `https://react.dev/`
[6] `https://nextjs.org/`



(a) Sequence diagram of dataset creation.



(b) Sequence diagram of mitigation algorithms.



(c) Sequence diagram of answering a question.

Figure 5.2: Detailed sequence diagrams describing interactions between system components when main domain events occur.

site generation. The component-based architecture fits well because the questionnaire can evolve, and the frontend must be arranged to be changed accordingly. A web-based technology has been chosen for several reasons: (i) cross-platform compatibility, (ii) ease deployment and maintenance, (iii) accessibility, (iv) powerful features offered by modern frameworks.

5.2 Testing

Testing is a fundamental part of the software development process, in this research-oriented software project it has been conducted following the philosophy of **Test Driven Development (TDD)** [Bec22]. This approach ensures that each implemented feature is tested thoroughly, having one or more tests that verify its correctness. Essentially, it consists of writing the tests for a feature before its actual implementation, followed by writing the implementation to make the tests pass. The entire testing environment has been set up using the unittest[7] framework in Python.

Unit Testing. Unit tests have been used to test backend main features such as general questions graph, project contexts modifications, and answer selection. These tests also cover serialization methods for domain entities and the correctness of main automation scripts. In fact, fairness computation performed by such scripts, are tested using mocked components. This testing environment ensures that the scripts work correctly regardless they are already integrated into the whole system or not.

Integration Testing. Integration tests helped to verify correctness of the whole system, ensuring that all components work together as expected. Primarily, the interaction between API, business logic, and database has been tested. Each endpoint has more than one test case, covering different scenarios and edge cases. The starting point is an HTTP request, that triggers business logic. The major part of these tests, involve database operations, for this reason, during test executions, a container with a Neo4j instance is started, and the database is initialized with

[7] `https://docs.python.org/3/library/unittest.html`

Package	Statements	Missed	Coverage
application	984	71	93%
domain	421	52	88%
infrastructure	801	69	91%
presentation	23	1	96%
resources	19	0	100%
utils	127	10	92%
TOTAL	**2375**	**203**	**91%**

Table 5.1: Test coverage report aggregated by main packages.

test data. When test suite execution is completed, the container is stopped and removed.

Integration tests do not involve just Rest API, but also the event-driven one. Indeed, interactions between automation scripts and backend have been tested too. These tests require a running Kafka instance, so, similarly to the database, a container with Kafka is started and stopped during tests execution.

Test Coverage. In table 5.1 is shown a report of the test coverage of the project. Test coverage is useful to understand how much code of a software artifact is actually covered by tests. Of course, the higher the coverage percentage, the better. Despite 100% of coverage is almost utopian, the 91% reached in the project is still a good result for a research-oriented software project.

5.3 Deployment

All components previously described are containerized and deployed using Docker[8]. In figure fig. 5.3 is shown the deployment diagram of the system. The configuration is set up to be easily deployed using one command line instruction with Docker Compose.

Persistence of data is guaranteed by volumes mounted in the containers, so data is not lost when containers are stopped.

[8] `https://www.docker.com/`

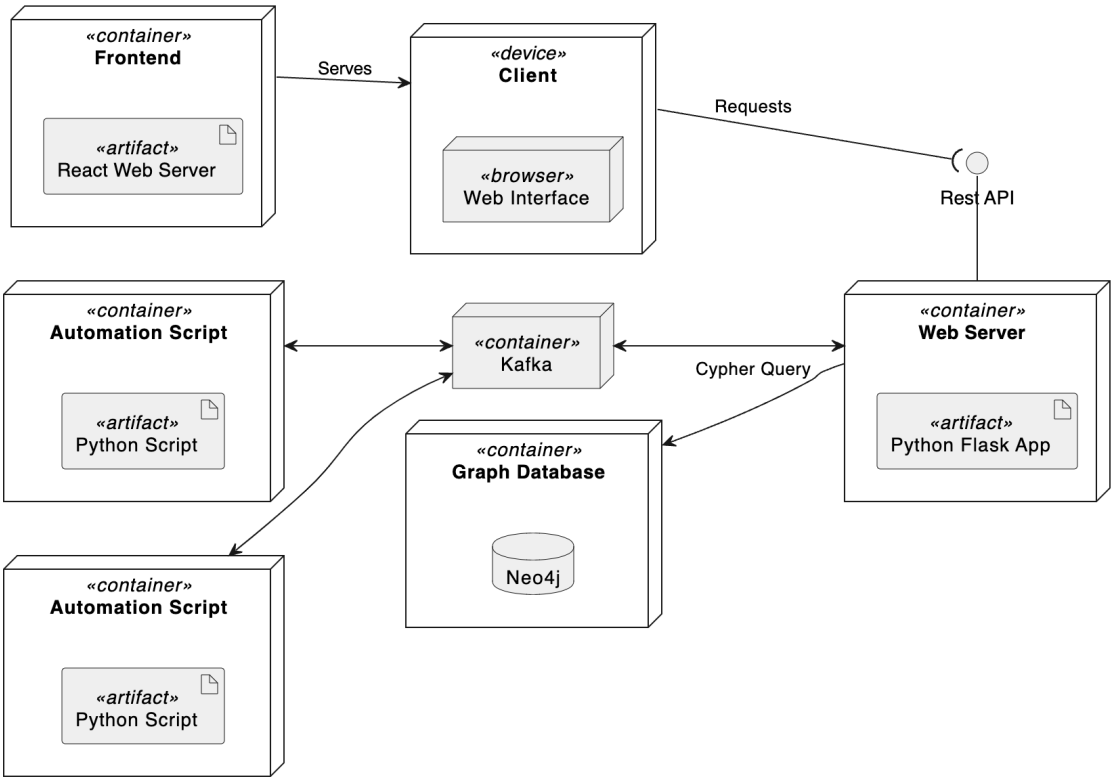


Figure 5.3: Deployment diagram of the system.

Finally, the system release deployed is the artifact produced through a Continuous Delivery (CD) pipeline which is successfully completed after all tests are passed.

Chapter 6

Validation

A contribution of this nature, in which are present both conceptual (the meta-methodology) and technical (the software artifact) parts requires a multi-faceted validation process. Validation is further complicated by the critical topic treated in this work, which is fairness in AI. This, in fact, makes it necessary to involve different stakeholders, such as ethicists, legal experts, and representatives of vulnerable groups, to ensure that the system is aligned with the principles of fairness.

6.1 Requirements Satisfaction

First, it is important to remark the requirements defined in section 3.1 and how they have been satisfied.

R1—Requirements Translation → Q/A Mechanism: the methodology involves socio-legal requirements translation into a simple and structured set of questions. A set of question designed for non-techcnical and non-legal people are more comprehensive than a set of legal constraints. Through this mechanism, it is also possible to practically constrain the development process depending on the paths designed in the questions graph.

R2—Context and Domain Awareness → Questions Design: through a well-designed set of questions, it is possible to incrementally collect information about the application domain and the cultural context in which the

system operates. Depending on the answers given by the BU, the system can adapt the flow of questions branching at the right points, providing tailored questions based on the context.

For instance, the answer to the question "In which area of application will the system be used?" will influence the following questions, leading to a questionnaire customization based on the application domain.

R3—Adaptability → Questions Graph: adaptability is intrinsically achieved by using a graph structure to represent the Q/A mechanism. In fact, it is possible to change the graph structure easily, adding, removing, or modifying questions and answers, creating new paths, and so on. This also enables a "versioning" mechanism, which effectively addresses a possible volatile context.

R4—Building the AI System → Software Reification: this contribution, rather than providing just an abstract specification of the methodology, provides also a guideline-provisioning software system usable directly by stakeholders. In the workflow, BU interacting with the system, can compile the questionnaire but at the same time, behind the scenes, backend processes operate to automate technical steps such as training and mitigation. In the end, the system will provide an AI system that has been subject to fairness considerations since the beginning of the development process.

6.2 Quality Assurance

From a technical point of view, the validation of development process is achieved by using DevOps practices. The constant develop of tests alongside core features (with TDD), allows applying Continuous Integration (CI) and Continuous Delivery (CD), benefiting of all the advantages that these practices bring.

Test Driven Development: The system has been developed following the TDD approach. TDD is a software development approach that emphasizes writing tests before writing the actual code. In this way, the developer is forced to think about the expected outcomes of the code before writing it.

By defining expected outcomes in advance, TDD helps prevent defects early, promotes cleaner and more modular code, and facilitates easier refactoring.

CI/CD: The system is continuously integrated and tested to ensure the codebase remains in a working state. A CI pipeline runs tests on each repository push, merging the code into the main branch if successful. This practice keeps the codebase stable, preventing new features from breaking existing ones. The system is automatically deployed once all tests pass. A CD pipeline builds the application, runs tests, and deploys it to production. This ensures the application remains up-to-date, making new features available as soon as they are developed.

Code Quality: To improve code quality, CI pipeline also runs code quality checks. These are performed using Black Formatter[1]. Black is a code formatter that automatically formats Python code according to a particular strict set of rules, without subjective style options.

6.3 Participatory Sessions

Fairness is one of the four "Ethical Principles in the Context of AI System" as outlined in the Ethics Guidelines for Trustworthy Artificial Intelligence (EGTAI) [EC19]. This principle is deeply interwoven with the other three—respect for human autonomy, prevention of harm, and explicability. A key pillar of EGTAI is *lawfulness*, and fairness is integral to numerous fundamental rights, laws, and governing principles, such as the European Pillar of Social Rights. This pillar serves as a guiding light for a fair and inclusive Europe, advocating for gender equality, equal opportunities, and access to essential services.

For these reasons, the validation of technologies and methodologies aimed at addressing fairness issues in AI must adopt a validation approach that begins with the co-creation and co-design phases of the methodologies and technologies themselves, using participatory design approaches. This approach was employed in the Aequitas process. Specifically, co-design and co-creation sessions were

[1] `https://black.readthedocs.io/en/stable/index.html`

conducted to embody socio-legal requirements within the technology. This led to the development of a meta-methodology presented in chapter 3, which was then validated in iterative co-creation sessions. The participatory approach involves setting up various focus groups, where diverse stakeholders convene to discuss, challenge, and refine the AI's requirements and functionalities. These stakeholders include AI developers, companies, representatives from vulnerable groups, legal experts, ethicists, and end-users. The objective is to incorporate a broad spectrum of perspectives to ensure the AI system is not only technically sound but also ethically aligned and socially beneficial. During these focus group interactions, each participant offers their expertise and insight, which are crucial for identifying potential biases and ensuring the AI system upholds fairness principles. Ethicists might highlight ethical dilemmas overlooked by developers, while legal experts ensure compliance with relevant data protection and anti-discrimination laws. The iterative nature of these co-creation sessions facilitates continuous feedback and enhancements, thereby fortifying the AI system against biases and aligning it more closely with the principles enshrined in the European Pillar of Social Rights. This method of validation through participatory design is essential for tackling the complex issues of fairness in AI, ensuring that the technology positively impacts society without perpetuating existing inequalities.

The feedback collected in the different workshops and focus group can be parsed to gather insights that are relevant to the scope of the AEQUITAS platform and what are the considerations it should be built upon. In particular, we analyzed the input provided by the underrepresented groups to extract desiderata and potential actions (technical and non-technical) that can guide the design and development of the tools and methodologies created throughout the project. These insights are collected in section 6.3.1.

6.3.1 Detailed Feedbacks

AI: Collaboration vs. Conflict.

Feedback: The relationship between humans and AIs can be either one of collaboration or conflict.

Insights: Clearly state that AI techniques are a tool in the hands of humans.

During the design process, stress that the user will be in charge of developing and deploying the AI systems, with the possibility of taking granular decisions concerning how the AI systems will be implemented and used. Do not build exceedingly automatized tools; in any case, provide a transparent description of the steps that have been automated.

Societal biases.

Feedback: Algorithms reflect societal biases.
Insights: Take bias into consideration during the design phase of new algorithms (development phases must be interspersed with bias detection actions applied to the data used as input and to the output of the developed tools). Provide bias detection mechanisms for both datasets and algorithms.

Diversity in the teams of programmers.

Feedback: It is important to diversify the team of programmers in charge of the AI system development and educate them to an inclusive mindset.
Insights: The design phase of the AI algorithm should involve people with a diversified technical/expertise background; this requirement is addressed by the meta-methodology presented in chapter 3, that explicitly asks for the inclusion of a diverse set of experts during the design process.

Dataset's Cleaning.

Feedback: Datasets should be cleaned, and content should be constantly controlled.
Insights: Great emphasis should be put on the bias analysis techniques, which should be easy to use and produce an output as easy to interpret as possible.

Intersectionality and Inclusivity.

Feedback: It is fundamental to consider intersectionality; and get out of one's own context and "privileged" situation.

Insights: Create diverse developing teams and involve people with different backgrounds. Explicitly include considerations of intersectionality during the development of new AI tools; this is aligned with the Q/A mechanism described in chapter 3—the key part is to include questions about intersectionality at the beginning of the questionnaire.

Awareness.

Feedback: Awareness among citizens is pretty low.
Insights: This is more a dissemination action; however, the system developed should be transparent and publicly available, thus allowing greater awareness.

AI Governance and Transparency.

Feedback: They are essential aspects of the process.
Insights: Use a transparent developing and deploying methodology. Strive to make as much code as possible public and well-documented, in order to increase transparency and auditability by government bodies.

Human oversight.

Feedback: Maintain human accountability for decisions made by AI systems; AI should assist, not replace, human judgment.
Insights: Enhance the feeling of user empowerment by developing a clear and transparent UI. During the design phase do not exceed with automated process but involve humans' feedback at every phase; explicitly foresee that the decision points will have to be taken care of by a human.

Recognition of Individual Normalcy.

Feedback: AI systems should not assume a statistical "normality" but instead recognize the normality of the individual. This involves understanding how AI might affect different minorities.
Insights: Move away from broad, one-size-fits-all statistical models and instead

develop models that take into account individual variability; implement personalized models. Incorporate sensitive attribute awareness where the AI model identifies different demographic groups (e.g., race, gender, disability status) and treats these attributes as critical factors in decision-making. Provide clear outputs for the bias detection and mitigation mechanism, to allow users or auditors to understand how specific features (e.g., minority status) influence the outcomes.

Data Disaggregation.

Feedback: Data should be disaggregated to consider all relevant aspects, such as ethnicity, migratory background, age, etc., rather than treating groups (e.g., women) as homogeneous.
Insights: The tools provided by AEQUITAS will allow measuring bias present in the data—and write clear and easy-to-understand reports. During the design phase, careful attention will be place on the selection of the right data to be used to train AI model, via explicit questions asked to developers to force them to consider if the data should be disaggregated of could be used as is (bias detection methods can be used to perform statistical test as well). Again, during the design phase, special care will be devoted in identifying users and other people and groups potentially affected by the AI system in development; the questionnaire pipeline serves to make sure that all relevant aspects will be considered.

Methodological Standards.

Feedback: There should be methodological standards, where AI is tested and refined based on real-world feedback.
Insights: Develop AEQUITAS tools using well-proven standard and well-known technologies. Aim at producing as much open-source code as possible. Clearly document all the methods developed and the underlying code to facilitate audition and evaluation.

All these requirements have been embodied in both the methodology and the experimental environment (for example, the experimental setting allows for data analysis and checks on distributions). Additional validation sessions that follow

participatory approaches and co-design methodologies have started, and the collection of incremental feedback is ongoing to continuously refine and enhance the methodologies and the experimental setup. These sessions utilize participatory approaches and co-design principles as well, ensuring comprehensive involvement from all stakeholders throughout the iterative development process. Feedback is collected from diverse sources, including real-time user interactions, expert reviews, and automated system analytics. This feedback is essential for identifying any potential issues related to data handling, algorithmic fairness, and overall system usability.

6.4 Software Assessment

A preliminary validation of the software has been achieved through its successful evaluation in the latest review of AEQUITAS project. During this assessment, positive feedback was received by reviewers regarding both the user experience and the underlying methodology, confirming their alignment with the project's objectives on AI fairness. This is an important outcome because provides an initial indication of the software's effectiveness in its objectives. Further validation steps will be pursued in future project reviews, where the software will represent a core deliverable of the project.

A usage example shown during the review, is a scenario where a BU wants to mitigate the `Adult` dataset[2] (see chapter A).

6.5 Limitations

With respect to the satisfaction of R1, it is defined *what* translated socio-legal requirements should be, but not exactly *how* to translate them. This still represents a complex collaborative challenge by a multidisciplinary team of experts, and it is not automatically addressed by the methodology. However, the methodology simplifies the process by providing itself some constraints. In other words, it forces the subjects involved in the requirements engineering to think about laws and legal

[2] `https://www.openml.org/search?type=data&sort=runs&id=179&status=active`

constraints in a pragmatic way, in order to translate them into a set of questions that can be answered by a person with no expertise.

Flexibility of questions graph, which satisfies R3, enables a *versioning* mechanism (see section 3.3) because changing the graph structure means creating a "new version" of the methodology. Thus, questions graph is open to changes and easily evolvable.

Conversely, questionnaires already started or completed, are affected by a version change, and it makes it difficult to adapt to new versions. The reason is due to the strong dependency between questionnaire (project-related graph) and general questions graph. In fact, each update to the questionnaire relies on the general graph, and so, general graph changes will lead to inconsistencies in the questionnaire. Therefore, in case of a version change, it is necessary to keep the oldest version to not break the support for the already started questionnaires. At the moment, the only way to adapt to new versions, is starting a new project (and relative questionnaire) from scratch.

A possible limitation respect to R4, is regarding the variety of AI systems that can be built. Indeed, software allows uploading datasets of any dimensions, and this could lead to possible issues in terms of computational resources. Theoretically, the methodology does not impose any constraint on the size of dataset or the complexity of the model, but in practice (using the software provided), it is necessary to consider the computational power available to the system.

Another important aspect is the time needed to perform fairness computations and model training, which can be a bottleneck in the system. This problem not only influences the backend system, but also affects the user experience, as the BU has to wait for the system to proceed correctly the questionnaire. Actually, computations performed by backend service and automation scripts are designed to be asynchronous to improves the user experience, but with the assumption that eventually—in a reasonable time—computations will terminate and user will be able to proceed with updated information. If computations take too long, BU could proceed with missing or outdated information, leading to possible misuses of the system.

Chapter 7

Conclusions

This thesis proposes a (meta-)methodology for building fair AI systems. The methodology does not offer a direct solution to all kinds of AI systems, but it provides a pragmatic way to define a custom development process for each specific system. The methodology is based on a Q/A mechanism, which helps to: (i) translate socio-legal requirements into a comprehensive set of questions, (ii) incrementally collect information about the application domain and the cultural context, (iii) adapt the development process to the specific needs of the system, (iv) build the fair AI system in the end. The methodology is designed to be flexible and adaptable, allowing itself to evolve and be refined over time, enabling the development of fair AI systems in different contexts. In addition, changes in the application domain can be easily addressed by creating new "versions" of the methodology, in other words, revisiting the questions graph structure modifying existing paths or creating new ones.

The methodology is reified into a software system, which provides a practical way to exploit the methodology in the development process. In particular, the software system is intended to be used by stakeholders, to guide them through the development process. In this way, Business User can interact with the system answering questions while, at the same time, the system automates technical steps such as training and mitigation.

Finally, the entire work has been validated considering both conceptual and technical aspects. For the technical aspect, the software system has been con-

stantly validated through DevOps practices including automated testing, release management, and code quality checks. For the conceptual side, the methodology has been validated through participatory design sessions, project reviews, and feedback from stakeholders and unprivileged groups.

7.1 Future Works

Future research will focus on both enhancing the software tool and refining the methodology. In particular, the topics require further improvements are outlined below.

- **Coverage of more use cases**: the methodology should account for both pre-existing and new AI systems. It is reasonable that the methodology should be able to be applied to new software systems, but it would be a big lack if it could not be applied to already existing systems, remembering that there are a lot of deployed systems that, probably, are not fair.
- **Research on the socio-legal requirements translation**: questions design has been conducted in a pragmatic way, but a more structured approach could be beneficial. A more structured approach could be based on a set of rules or guidelines to follow when translating requirements into questions.
- **Improve automation scripts**: at the moment, automation scripts perform consolidated computations such as fairness metrics and mitigation. In future works, it will be necessary to add new scripts to cover more technical steps and to improve the user experience.

Appendix A

Graphical User Interface

Images reported in this appendix, refer to the graphical user interface provided by frontend component described in chapters 4 and 5. These, are the main views showed to the stakeholder when he interfaces to the software tool to create an AI system. Note that this is just one of the possible sequences of images (paths of questions), as explained in chapter 3. In this case, the flow is representing the use case of a dataset mitigation.

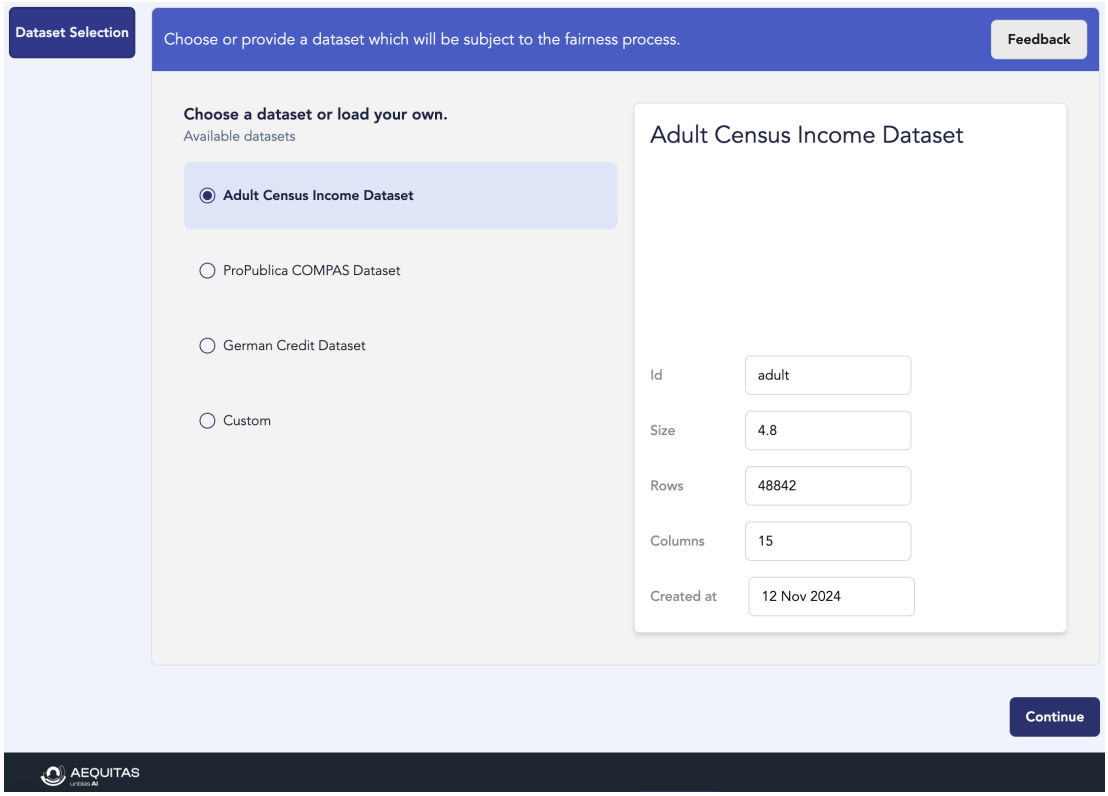


Figure A.1: Dataset selection view.

Figure A.2: Dataset view.

Figure A.3: Features selection view.

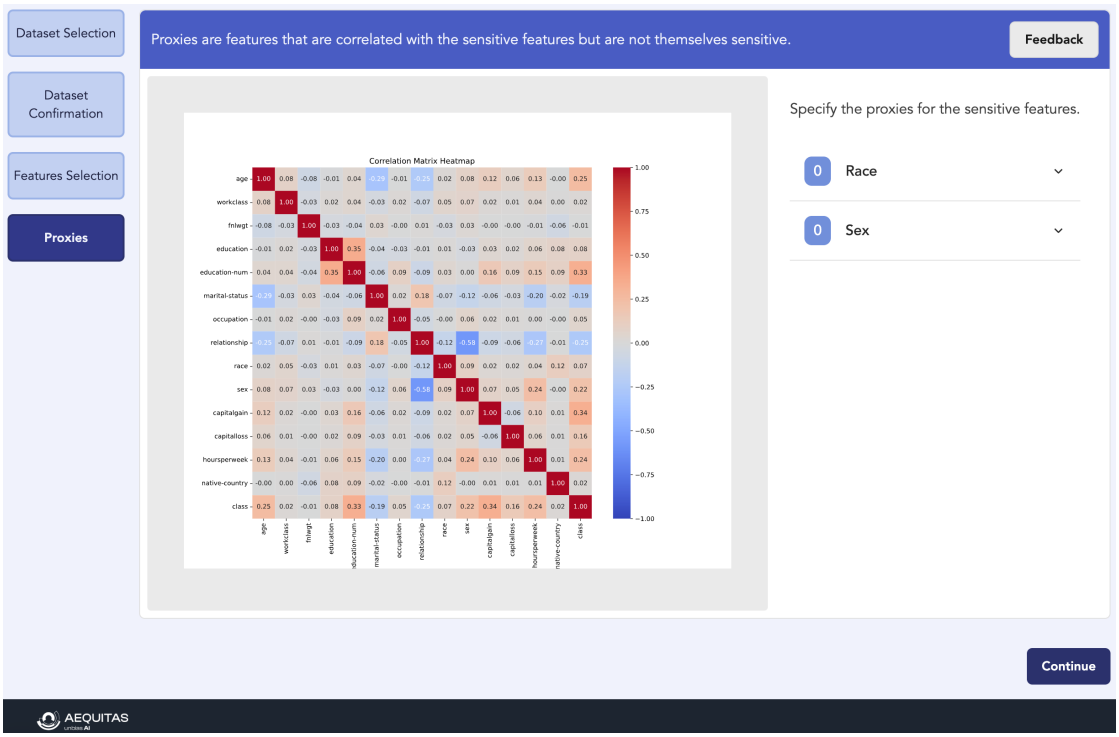


Figure A.4: Proxies view.

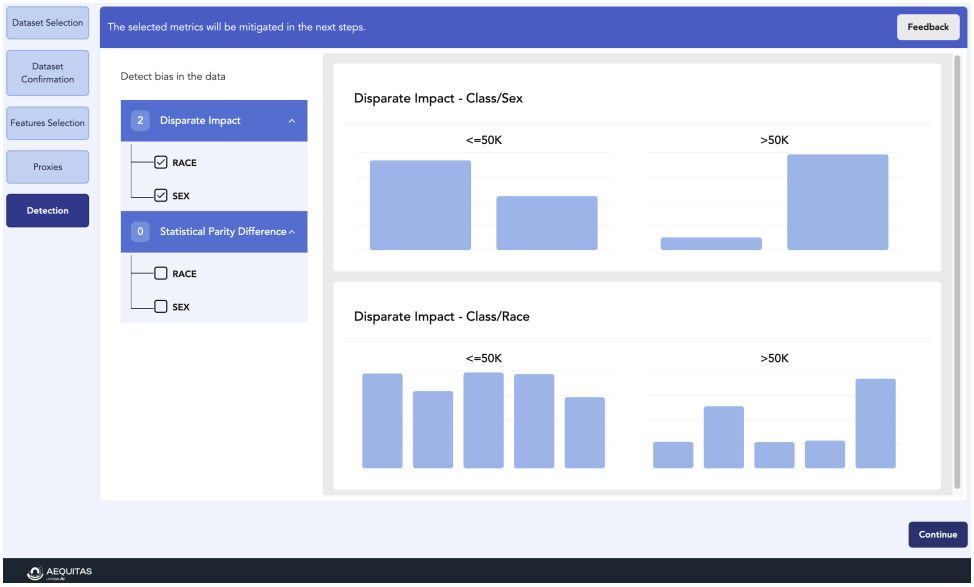


Figure A.5: Detection view.

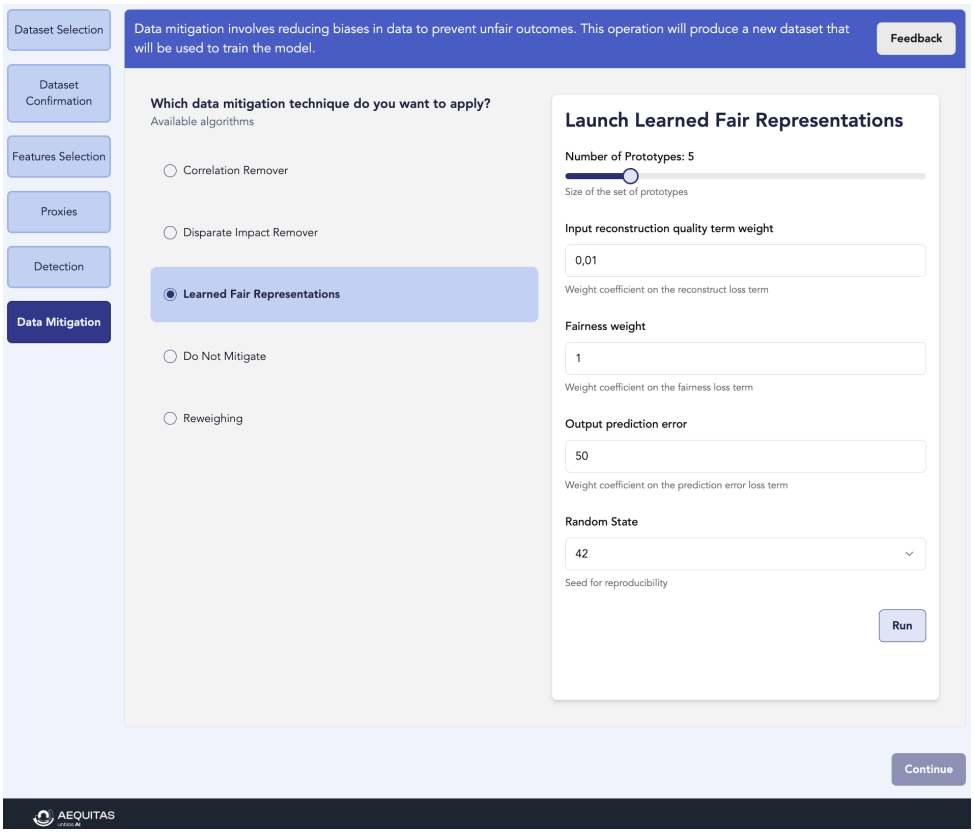


Figure A.6: Data mitigation algorithms selection.

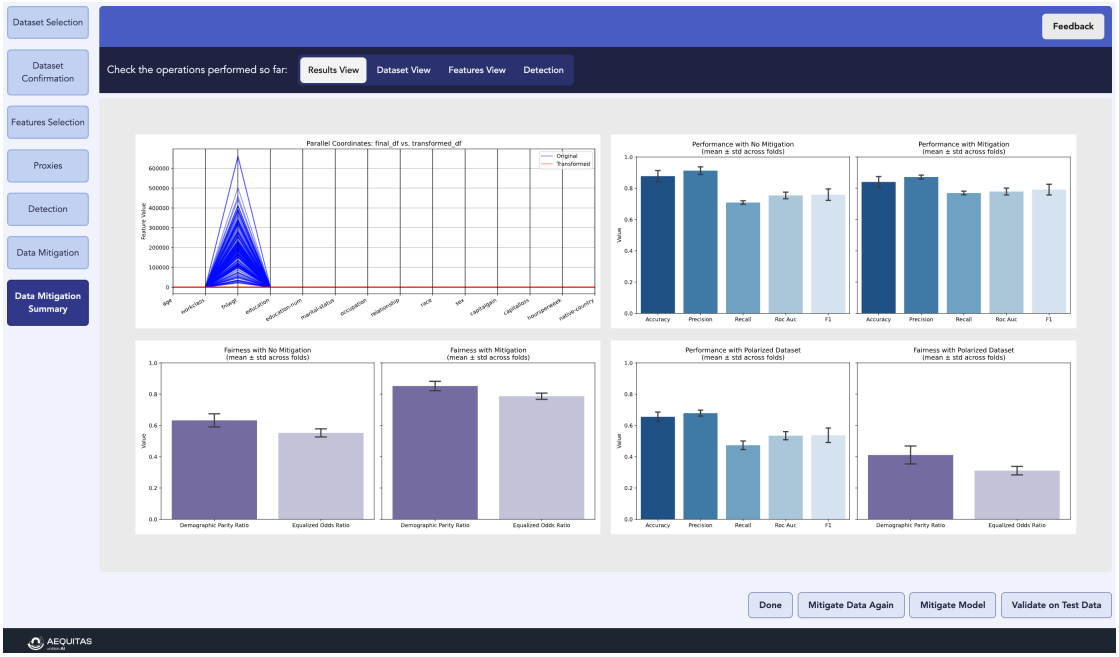


Figure A.7: Data mitigation results.

Bibliography

[Bec22] Kent Beck. *Test driven development: By example*. Addison-Wesley Professional, 2022.

[CCMO23] Roberta Calegari, Gabriel G. Castañé, Michela Milano, and Barry O'Sullivan. Assessing and enforcing fairness in the AI lifecycle. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, pages 6554–6562. ijcai.org, 2023.

[CMS+25] Giovanni Ciatto, Mattia Matteini, Laura Sartori, Maria Rebrean, Catelijne Muller, Andrea Borghesi, and Roberta Calegari. AI-fairness: the FairBridge approach to practically bridge the gap between socio-legal and technical perspectives. In *Proceedings of the 58th Hawaii International Conference on System Sciences*, page 6499, 2025. (In press).

[DHP+11] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard S. Zemel. Fairness through awareness. *CoRR*, abs/1104.3913, 2011.

[DVN+22] Roxana Daneshjou, Kailas Vodrahalli, Roberto A. Novoa, Melissa Jenkins, Weixin Liang, Veronica Rotemberg, Justin Ko, Susan M. Swetter, Elizabeth E. Bailey, Olivier Gevaert, Pritam Mukherjee, Michelle Phung, Kiana Yekrang, Bradley Fong, Rachna Sahasrabudhe, Johan A. C. Allerup, Utako Okata-Karigane, James Zou, and Albert S. Chiou. Disparities in dermatology ai performance on a

diverse, curated clinical image set. *Science Advances*, 8(32):eabq6147, 2022.

[EC19] High-Level Expert Group on Artificial Intelligence European Commission. Ethics guidelines for trustworthy ai, April 2019.

[EPL23] Francesca Fallucchi Erasmo Purificato, Flavio Lorenzo and Ernesto William De Luca. The use of responsible artificial intelligence techniques in the context of loan approval processes. *International Journal of Human-Computer Interaction*, 39(7):1543–1562, 2023.

[Fer24] Emilio Ferrara. Fairness and bias in artificial intelligence: A brief survey of sources, impacts, and mitigation strategies. *Sci*, 6(1), 2024.

[FFM+15] Michael Feldman, Sorelle A. Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. Certifying and removing disparate impact. In Longbing Cao, Chengqi Zhang, Thorsten Joachims, Geoffrey I. Webb, Dragos D. Margineantu, and Graham Williams, editors, *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, pages 259–268. ACM, 2015.

[IML23] Zahid Irfan, Fergal McCaffery, and Róisín Loughran. Evaluating fairness metrics. In Ludovico Boratto, Stefano Faralli, Mirko Marras, and Giovanni Stilo, editors, *Advances in Bias and Fairness in Information Retrieval - 4th International Workshop, BIAS 2023, Dublin, Ireland, April 2, 2023, Revised Selected Papers*, volume 1840 of *Communications in Computer and Information Science*, pages 31–41. Springer, 2023.

[JMCB22] Jean-Marie John-Mathews, Dominique Cardon, and Christine Balagué. From reality to world. a critical perspective on ai fairness. *Journal of Business Ethics*, 178(4):945–959, 2022.

[Jos24] Jeena Joseph. Predicting crime or perpetuating bias? the ai dilemma. *AI & SOCIETY*, pages 1–3, 2024.

[KBP+24] Marten H. L. Kaas, Christopher Burr, Zoe Porter, Berk Ozturk, Philippa Ryan, Michael Katell, Nuala Polo, Kalle Westerling, and Ibrahim Habli. Fair by design: A sociotechnical approach to justifying the fairness of ai-enabled systems across the lifecycle, 2024.

[Mad21] Tambiama Madiega. Artificial intelligence act. *European Parliament: European Parliamentary Research Service*, 2021.

[Mar17] R.C. Martin. *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Robert C. Martin series. Prentice Hall, 2017.

[MT15] Scott Millett and Nick Tune. *Patterns, principles, and practices of domain-driven design*. John Wiley & Sons, 2015.

[RFV24] Carlotta Rigotti and Eduard Fosch-Villaronga. Fairness, ai & recruitment. *Computer Law & Security Review*, 53:105966, 2024.

[UKF+24] Daiju Ueda, Taichi Kakinuma, Shohei Fujita, Koji Kamagata, Yasutaka Fushimi, Rintaro Ito, Yusuke Matsui, Taiki Nozaki, Takeshi Nakaura, Noriyuki Fujima, et al. Fairness of artificial intelligence in healthcare: review and recommendations. *Japanese Journal of Radiology*, 42(1):3–15, 2024.

[WYBM20] Mark Weber, Mikhail Yurochkin, Sherif Botros, and Vanio Markov. Black loans matter: Distributionally robust fairness for fighting subgroup discrimination, 2020.

Acknowledgements

Optional. Max 1 page.