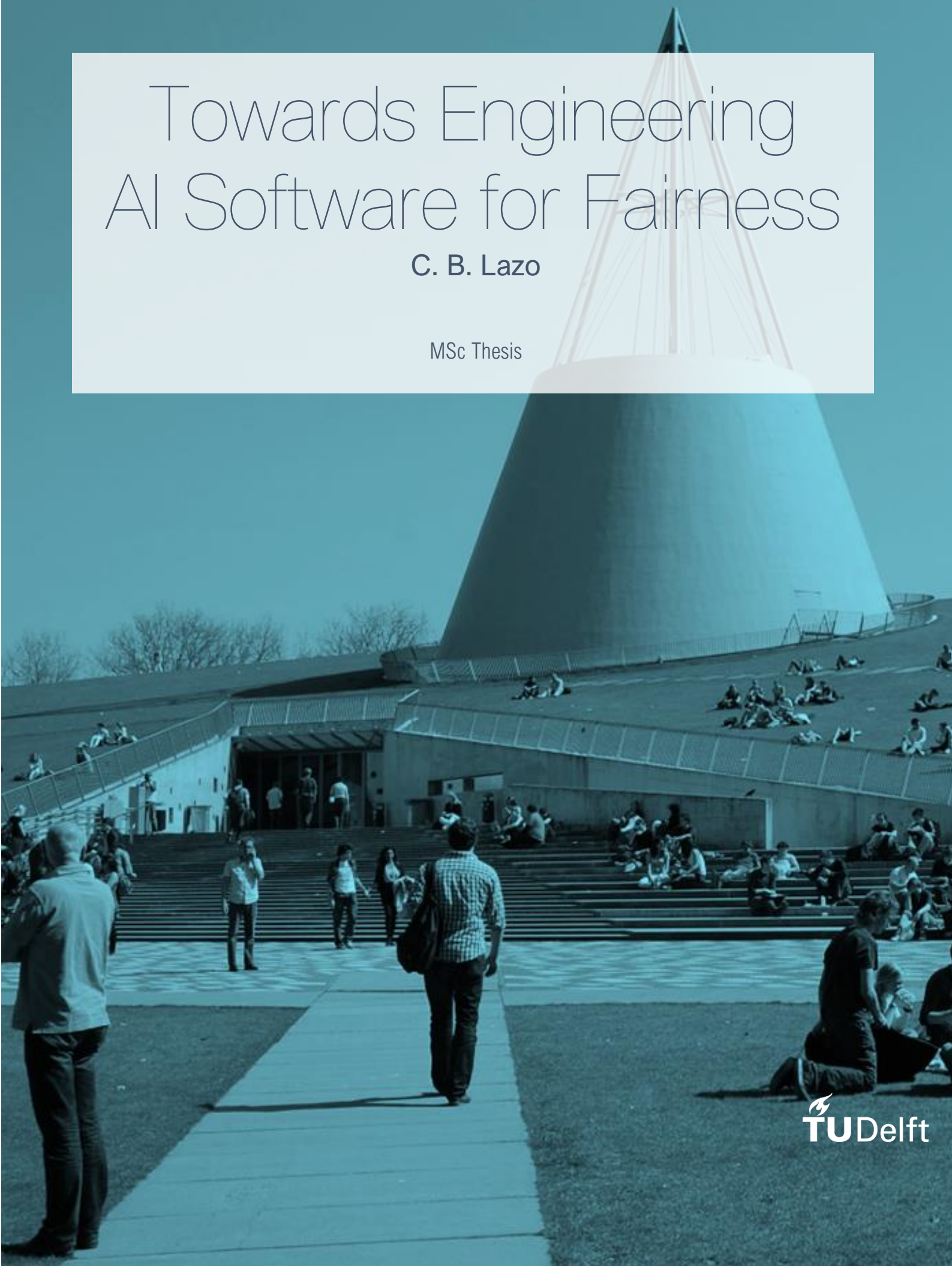
Towards Engineering AI Software for Fairness

C. B. Lazo

MSc Thesis



TU Delft

Towards Engineering AI Software for Fairness

A framework to help design fair, accountable and transparent algorithmic decision-making systems

by

Claudio Lazo

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday August 25, 2020 at 12:00.

Student number 4108833
Project duration September 1, 2019 – August 25, 2020

Thesis committee

Chairperson	Prof.Dr.Ir. G.J.P.M. Houben,	Web Information Systems, EEMCS, TU Delft
Supervisor	Dr. C. Lofi,	Web Information Systems, EEMCS, TU Delft
	Dr. R.R. Venkatesha Prasad,	Embedded Networked Systems, EEMCS, TU Delft

TU Delft

Abstract

Algorithmic decision-making (ADM) is becoming increasingly prevalent in society, due to the rapid technological developments in Artificial Intelligence. ADM make substantially impactful decisions about people: diagnosing whether we have a disease, what news and which ads we get to see, whether we are eligible for a job, benefits, a college or a loan, they show us personalized media and news, and steer the car that drives us home. However, ADM brings about ethical, legal and social issues by inheriting and perpetuating human biases, learning to discriminate—even learning gender or racial stereotypes, and lacking transparency and accountability. This unexpected and biased behaviour arises because these software systems are usually built without the specification of fairness requirements (i.e. what fair behaviour is expected of the system). We envision a Software Engineering for Values (SEfV) method that solves this problem. This study addresses that specification problem, aiming to help practitioners design ADM software for fairness. Using literature in social sciences—specifically organizational justice—the human value of fairness has been conceptualized in regard to ADM. This resulted in a *fairness tree* with four dimensions (procedural, distributive, informational and interpersonal fairness), which is further specified into 36 fairness norms. Subsequently, the fairness tree is related to current measures of fairness and techniques. Finally, we put forward the Software Engineering for Values (SEfV) framework, based on the principles of Software Engineering and Design for Values, and show how it can be applied to design ADM for fairness.

Experiments were conducted where participants ($n = 12$) performed a design task ($M = 3,75$ requirements specified) and an audit task for a hypothetical loan decision system—using a prototype of the SEfV framework. Participants found the prototype useful for both design as auditing, especially as a tool for reflecting on fairness considerations. This suggests that a high fidelity version would be useful for practitioners.

Keywords fairness; discrimination; bias; algorithmic decision-making; machine learning; software engineering; requirements engineering; Design for Values; AI ethics

Acknowledgments

This thesis project was the greatest challenge I got to experience during my time as a student. I loved that my thesis was multi-disciplinary. It was extremely interesting learning from so many different disciplines: philosophy, psychology, economics, sociology, while also refreshing my Software Engineering skills.

But I could not have done this alone, so I would like to make some acknowledgements. My gratitude goes out to Christoph Lofi and Agathe Balayn for guiding me through this venture into research. It was a pleasure to work with you. You were wonderful supervisors! To Agathe, I wish you good luck with the remainder of your PhD work.

I would like to thank Evgeni Aizenberg and Luciano Siebert for their input and involvement in the beginning, back when I had to venture into a field that I was not yet familiar with. The AiTech agoras were very inspiring and I enjoyed them very much. Also my thanks to Luis Cruz.

To all those that participated in my experiments, thank you for your interest and enthusiasm. I was very excited to share my work with you.

Finally, I would like to thank those that gave me moral support during the harder times of the thesis project. To Michel Rodrigues and the thesis support team, many thanks for the weekly reflections. I couldn't have done this without Liam, who gave me the emotional support when times got tough. Thanks to Daan and Liam for proofreading my thesis, and Joris for testing my prototype. Finally, a great big thank you to my parents, Oscar & Janine, who supported me in my adventure here at TU Delft.

C. B. Lazo
Delft, August 2020

Contents

List of Figures

List of Tables

1 Introduction

Algorithmic decision-making is becoming increasingly prevalent throughout society, with a substantial impact on human lives. Algorithmic decision-making (ADM) is defined as ”computerized implementations of algorithms, including those derived from machine learning or other data processing or artificial intelligence techniques, which are used to make or assist in making decisions” (Stoyanovich, 2019). This expansion is driven by the availability of large amounts of data of our behaviours (Monteith & Glenn, 2016) and rapid advances in many areas of Artificial Intelligence (AI) such as Deep Neural Network architectures (Rahwan, 2018). ADM is applied throughout public and private sector, making high impact decisions about individuals. For instance, in web information systems such as online job assessments, ad delivery, matchmaking, media recommendation, personalized news and personal assistants (Karanasiou & Pinotsis, 2017), but also risk assessments in banking and insurance, diagnosis and treatment in health care (Harris & Davenport, 2005). In the public sector ADM is used for law enforcement, criminal justice, benefit eligibility, edcuation, licencing (Citron, 2008), and immigration (Molnar & Gill, n.d.).

The application of Machine Learning exacerbates the benefits of ADM. Whereas humans have many cognitive biases that impede their decision-making (Arnott, 1998), automated systems are able to make precise, consistent, objective, decisions efficiently (Sheshasaayee, 2017). In Machine Learning (ML), a computer system is trained to recognize patterns in data. Seemingly outperforming humans, it is widespread belief that ML increases the quality of decisions (De-Arteaga et al., 2020). However, algorithmic decision-making appears to show the same symptoms as human decision-making (Molnar & Gill, n.d.).

ADM bring about ethical, legal and social issues such as bias, discrimination, a lack of transparency and accountability (Lepri et al., 2018). Human biases in data are easily learned and amplified by ML systems (Barocas & Selbst, 2016), text corpora contain stereotypes (Bolukbasi et al., 2016), more intelligent system will also be more opaque and make inexplicable decisions (Karanasiou & Pinotsis, 2017), and ubiquitous ADM may lead to exclusion of vulnerable groups such as individuals with mental illness (Monteith & Glenn, 2016). There are many examples of these issues: COMPAS, a recidivism risk assessment tool, was found to exhibit race-related issues (Verma & Rubin, 2018). Google’s image recognition system mislabelled an African American couple as gorillas[1]; Youtube automatically demonetizing videos that contain LGBT terms[2]; In the Netherlands, SyRI (*Systeem Risico Indicatie*), a risk indication system employed by Dutch public services raised numerous issues, among which a lack of transparency, bias and discrimination and privacy invasion (Timan & Grommé, 2020).

1.1. Problem statement

We make two observations that explain the unexpected, biased behaviour from the examples: First, practitioners do not design ADM structurally. ADM systems are predominantly developed analytically rather than through the structural method of Software Engineering, relying heavily on data analysis

[1] https://www.theverge.com/2018/1/12/16882408/google-racist-gorillas-photo-recognition-algorithm-ai
[2] https://www.theverge.com/2018/6/4/17424472/youtube-lgbt-demonetization-ads-algorithm

and model fitting on trial-and-error basis (Sculley et al., 2015). As ML systems require a vast, complex infrastructure, possibly introducing risks, the lack of Software Engineering in these systems may incur long-term costs (Sculley et al., 2015).

Second, practitioners do not design ADM for fairness. In order to design fair machine learning, practitioners face an upfront set of ethical challenges (Binns, 2017). In general, ML systems are built with a focus on efficiency and accuracy, and software design decisions are mainly ignorant of values (Mougouei et al., 2018). Even practitioners who are motivated to address fairness, face technical and organizational barriers (Holstein et al., 2019). As a result, unexpected and biased behaviour arises (Brun & Meliou, 2018).

To that end, the Software Engineering community has had multiple calls to action. Multiple research roadmaps ascertain the lack of methods to account for fairness (Brun & Meliou, 2018)—as well as ethics and values in general (Aydemir & Dalpiaz, 2018; Mougouei et al., 2018). Similar research challenges are identified from the ML practitioner's perspective (Holstein et al., 2019) and within database management systems (Balayn et al., 2020). Furthermore, within the ML fairness community there is a call for translating high-level notions (Passi & Barocas, 2019), choosing between notions of fairness (Binns, 2017; Verma & Rubin, 2018), and understanding the mapping between human notions of fairness and design practices (Zhu et al., 2018). **In essence, there is need for a method to design software for values.**

The field of *Design for Values* (DfV) is a potential source for such a method Software Engineering to account for values. DfV is a collection of design methodologies for the explicit translation of moral and social values into context-specific design requirements (van den Hoven et al., 2015). However, no existing method provides an integration of these activities (van de Poel, 2015b).

DfV does, however, provide criteria that such a method should support (van de Poel, 2015b):

1. *Discovery* and elicitation of values which are potentially worth pursuing,
2. *Specification* of general values in terms of design requirements,
3. *Translation* of such requirements into engineering characteristics,
4. *Choice* in design and resolving conflicts between criteria or values, and
5. *Verification* of the proposed design with respect to their incorporated values.

We envision a Software Engineering for Values (SEfV) method—–adapted to the criteria given by DfV—–that can provide guidance in the reflection on abstract values such as fairness, help with the structural specification, implementation and testing of ethical requirements; help choose among conflicting solutions and assumptions—all while improving transparency and accountability by explicating design decisions.

1.2. Research goal

The aim of this study is to help practitioners design algorithmic decision-making (ADM) software for *fairness*, by

1. developing the envisioned Software Engineering for Values (SEfV) framework; and
2. demonstrating and evaluating the application of SEfV to engineer ADM software for *fairness*.

The SEfV framework shall support the *specification*, *translation* and *choice* activities, leaving the rest for future work. We only consider the value *fairness*, meaning that we can leave out the *discovery* activity. Our reasoning behind this is that for the fair-ML community the value of fairness is the main focus, while for the Responsible AI community, fairness is a suitable example of a highly context-dependent abstract value. For eliciting other values, DfV methodologies such as Value Sensitive Design (Friedman et al., 1987) would be suitable. The *verification* activity is left out since this study focuses on the translation of fairness into the design. For further exploration of the verification activity, we refer to Aydemir and Dalpiaz (2018).

1.3. Research questions

In order to achieve the research goal (section 1.2), this study addresses four research questions.

RQ 1 requires us to conceptualize fairness. Its formulation is based on Aydemir and Dalpiaz (2018). In RQ 2, we relate the state of the art fairness measures and interventions to our fairness conceptualization. We based this research question on Mougouei et al. (2018). For RQ 3 (constructing the framework) and RQ 4 (demonstrating and evaluating the framework) we need to apply a design methodology to construct the SEfV framework. These research questions have been formulated based on Thuan et al. (2019, p.350).

RQ 1 — How can we conceptualize fairness in algorithmic decision-making (ADM)?

1.1. What prior knowledge about fairness is available in existing texts related to values?
1.2. How can we conceptualize fairness for ADM based on this prior knowledge?
  1.2.1. What is an adequate synthesis of prior knowledge about fairness?
  1.2.2. How does the conceptualization apply to ADM?

Approach to RQ1 We perform a literature survey in the disciplines that are relevant for defining fairness: philosophy, law, economics, sociology and psychology. For answering 1.2, we relate the factors of fairness to other notions of fairness, and to the ADM context.

RQ 2 — How do software design practices embed aspects of fairness?

2.1. What prior knowledge is available about software design practices for ensuring fairness?
2.2. How are these software design practices related to our articulation of fairness?

Approach to RQ2 A literature survey in relevant journals about Machine Learning, Data Mining, Knowledge Engineering, Neural networks, Fair-ML, as well as a number of seminal papers outside those journals.

RQ 3 — What is an adequate solution for the Software Engineering for Values (SEfV) framework?

3.1. What are the essential objectives and requirements for designing SEfV?
3.2. How can we elaborate SEfV to be compliant with the set of requirements?
3.3. How can we implement SEfV?

Approach to RQ3 We apply Design Science Research Methodology (Peffers et al., 2007) to answer 3.1, 3.3 and 3.4. In order to answer 3.2, we perform a literature survey within the Software Engineering and the Design for Values literature for related work (section 2.3).

RQ 4 — To what extent does the application of SEfV support translating fairness to ADM design?

4.1. How can we use SEfV to translate fairness to ADM design?
  4.1.1. What are the design parameters of an ADM systems?
  4.1.2. How does SEfV translate fairness to ADM design?
4.2. How can we evaluate SEfV?

Approach to RQ4 Similar to RQ3, we apply Design Science Research Methodology (Peffers et al., 2007) to answer 4.1 – 4.2. For question 4.1.1 we perform a literature survey for general ADM and DSS descriptions.

1.4. Contributions

The main contribution of this work is the envisioned **Software Engineering for Values framework**, both a conceptual model and an instantiation of the framework (in order to apply SEfV to translate the value *fairness* to *ADM* software design). As a side contribution, we also introduce a **general architecture for ADM systems**, extending the Decision Support System model of Power (2002) with a decision-making component. This architecture may act as a template for designing ADM software.

Another contribution of this work is a **conceptualization of fairness in ADM**, that we have related to Fair Machine Learning design practice. This conceptualization, is a tree that specifies fairness into a set of norms that generally should apply, and may contribute to the formulation of AI standards. As a side

contribution, we identify potential **research gaps**, namely the aspects of our fairness conceptualization that are not yet addressed in Fair Machine Learning design practice and thus be investigated.

This document is further structured to address the research questions in a logical order:

- In *chapter 2* we lay out the background on software engineering and design for values, and research that is similar to ours.
- In *chapter 3* we synthesize a conceptualization of fairness (RQ1).
- *Chapter 4* describes the fairness measures and interventions within Computer Science and how they relate to our notion of fairness (RQ2).
- *Chapter 5* describes our proposed SEfV framework and the design science research methodology (RQ3).
- We show how SEfV is used to translate fairness to ADM design in *chapter 6* (RQ4.1).
- We evaluate the SEfV framework in *chapter 7* (RQ4.2).
- For discussion of the result and our final conclusions and recommendations, we refer to *chapter 8*.

2

Background & related work

This chapter provides the background on Software Engineering (section 2.1) and Design for Values (section 2.2) that is necessary for the rest of this study, and describes works that are related to our study (section 2.3).

2.1. Software Engineering practices

Software Engineering is the *"application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software"* (p.29), and generally consists of the stages: Requirements Engineering, Design, Implementation, Testing and Maintenance (IEEE Computer Society, 2014). Requirements engineering establishes "what" the software should do (section 2.1.1), design establishes "how" the software does that (section 2.1.2). The stages implementation, testing and maintenance — however important in SE practice — are not relevant within the scope of this study.

2.1.1. Requirements Engineering

Requirements Engineering (RE) is the systematic handling of software requirements (IEEE Computer Society, 2014). Software requirements explicitly express needs and constraints that apply to a software product. The Software Engineering Body of Knowledge (SWEBOK) specifies the following four activities: requirements elicitation, requirement analysis, requirement specification and requirement validation. As shown in figure 2.1, these are sequential activities, but have an iterative nature.

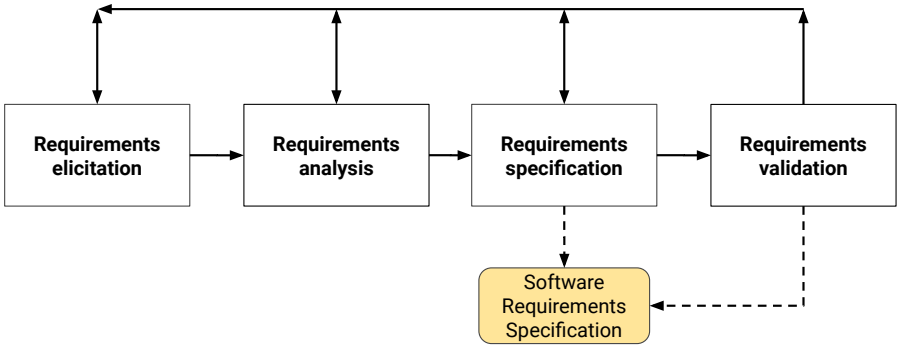


Figure 2.1: The general Requirements Engineering process (solid arrows represent activity flows, dashed arrows represent information flows).

Elicitation Requirements elicitation is concerned with origins of software requirements and how these can be collected. Requirements can come from software goals, domain knowledge, stakeholders, business rules and the operational/organizational environment. The most common elicitation techniques and approaches in RE are interviews, domain analysis, groupwork, ethnograpy, prototyping, goal anal-

ysis, scenarios or user stories ("*As a* `<role>`*, I want* `<goal/desire>`*, so that* `<benefit>`"), and viewpoints (Zowghi & Coulin, 2005).

Analysis Requirement analysis is concerned with creating a consistent set of requirements, discovering how the software must interact with its environment, and to elaborate requirements for the software artifact. The stages are classification, negotiation and optionally formal analysis.

With *requirements classification*, the requirements are provided with extra classifications in order to make informed trade-offs during requirements negotiation. The most common classifications are: functional/non-functional requirement, derived/emergent, product/process, priority, scope and volatilty. Besides classifications, requirements have attributes such as a unique identifier, the verification method, rationale, source, and change history.

During *requirements negotiation*, conflicts between requirements are detected and resolved. Stakeholders may have incompatible needs, but trade-offs with the stakeholders are necessary to reach a consensus. Requirements prioritization is a (collaborative) way to resolve conflicts and make necessary trade-offs (Aurum & Wohlin, 2005). From simple to sophisticated the most common prioritization technique are: top-ten requirements, numerical assignment, ranking, 100-dollar test, cumulative voting, Analytical Hierarchy Process (AHP). it is advised to use the simplest appropriate prioritization technique (Berander & Andrews, 2005).

Optionally, *formal analysis* is useful at a later, more concrete, stage. It consists of formally expressing requirements in a language with formally defined semantics (e.g. predicate logic).

Specification Requirements specification refers to the production of a document that can be systematically reviewed, evaluated, and approved. This document is the software requirements specification (SRS).

Validation During requirements validation, the quality of the developed requirements is reviewed, according to predetermined quality criteria such as consistency and correctness (Pohl & Rupp, 2015). The goal is to the right software will be designed (validation) and that the software is designed right (verification).

Validation is concerned with ensuring the software is what the stakeholders expect. Techniques are: requirement review, prototyping, model validation and acceptance tests.

During *verification*, the SRS is checked whether it conforms to company standards and that it is understandable, consistent and complete (IEEE Computer Society, 2014). Verification also refers to ensuring that requirements are verifiable: Requirements are described such that tests or measurements can prove that the software satisfies the requirement (Pohl & Rupp, 2015).

2.1.2. Software Design

While Requirement Engineering describes what is needed from the software (SRS), the software design phase explicates what should be built. The software design describes the software implementation in terms of diagrams and models, allowing to evaluate with stakeholders whether the software design fulfills the requirements. Models may describe the software architecture, components, interfaces, and other characteristics of a system or component" (IEEE Computer Society, 2014).

Out of the many different modeling languages and techniques, the Unified Modeling Language (UML) has become the standard. Some languages and techniques specialize in, for example, the modeling of processes (Business Process Modeling Notation, data flow diagrams) or functions (IDEF), while others are general modeling languages for the entire enterprise architectures (ArchiMate). UML is a general-purpose modeling language, that has been adopted as an ISO standard[1]. Using UML, a graphic description of software activities, components, interactions, external interfaces (Rumbaugh et al., 2005).

2.2. Design for Values

Design for Values (DfV) is a methodological design approach that aims at making moral values part of technological design, research, and development (van den Hoven et al., 2015). Values are defined

[1] https://www.iso.org/standard/32620.html

as *"what is important to people in their lives, with a focus on ethics and morality"* (Friedman & Hendry, 2019). The main motivation of DfV is to mitigate value failures of design, embodiment of values in the design, and generation of values through design. DfV is closely linked to Value Sensitive Design (VSD) and participatory design.

Participatory design refers to design practices that involve the end users as co-designers, making it more likely that the final result of the design process will represent the values of the end users van der Velden and Mörtberg (2015a). Thus, values have been integrated mostly through participatory design methods (Barn et al., 2015).

Value sensitive design (Friedman et al., 2008; Friedman & Hendry, 2019; Friedman & Nissenbaum, 1996) is a design theory that rests on a set of commitments that clarify the human relationship with technology. It stipulates that the design of technology should be sensitive to human values. Some values often implicated in system design are human welfare, privacy, freedom from bias, informed consent, accountability, trust, universal usability (Friedman et al., 2008).

VSD iteratively integrates three types of investigations that inform each other (Friedman & Hendry, 2019):

- **Conceptual investigations** are theoretical explorations of the key issues and concepts at stake. These are the stakeholders (both direct and indirect), the values at stake, and their conceptualization.
- **Empirical investigations** consists of identifying the stakeholder's needs, views and experiences in relation to the technology and the value it implicates.
- **Technical investigations** evaluate how the technology supports or inhibits the values and norms elicited from the conceptual and empirical investigations.

The proposed Software Engineering for Values (SEfV) framework adopts the DfV design approaches, so it is necessary to explore the characteristics in which DfV differs from Software Engineering. DfV characterizes itself by a socio-technical view of design rather than purely technical, providing guidance in the involvement of stakeholders, resolving conflicts between values, and the translation of values to design requirements. In the following sections, we explore those characteristics.

2.2.1. Identifying and involving stakeholders

Both conceptual and empirical investigations relate the technology to stakeholders and their values. Stakeholder analysis is a particularly important method of VSD as it identifies **direct** stakeholders, those who directly interact with the technology or its output) as well as **indirect** stakeholders, those who are impacted or want to make an impact on the technology. These stakeholders may be broadly defined and the designer should take into account that individuals may belong to multiple stakeholder groups or subgroups, thus to consider roles (e.g. user, operator, auditor) than individual positions (Friedman & Hendry, 2019, p.64). For example, an urban planner who lives in the area that she is developing plans for (Friedman et al., 2008).

In DfV, stakeholders' potential harms and benefits from the technology are elicited, as well as which values are important to them (Friedman et al., 2008). Since empirical investigations (engaging with stakeholders) relate to human activity, all quantitative and qualitative research methods in social sciences may apply to VSD (Friedman et al., 1987). There are numerous techniques for eliciting values and requirements from stakeholders: interviews, surveys, envisioning cards, value scenarios and participatory prototyping (Friedman & Hendry, 2019), ethnography, participant observation, and longitudinal case studies (van der Velden & Mörtberg, 2015b).

Despite stakeholder analysis being a part of the requirements elicitation process, it does not correspond to stakeholder analysis within DfV. In regular SE practices, stakeholder analysis refers to eliciting requirements from the client and users of a system with respect to achieving the system's goal. Contrarily, in DfV, the same task identifies a broader set of stakeholders and—more importantly—their values with respect to the software system and its context.

2.2.2. Value conflicts

Value conflicts arise when different values prefer different design options (van de Poel, 2015a). As *"[t]ouching one value implicates others"* (Friedman & Hendry, 2019, p.44), there lies a challenge in

dealing with these tensions between values. For instance, improving privacy by using less personal data might result in higher unfairness (Corbett-Davies et al., 2017).

There are numerous strategies to handle value conflicts. When values are regarded as criteria for choosing among different design options, methods are cost-benefit analysis, direct trade-offs, maximin, satisficing, respecification, and innovation van de Poel (2015a). With cost-benefit analysis, all values are expressed in monetary terms, and the best option is chosen. Direct trade-offs means that we trade in a loss on one value scale, for a gain in another scale. Maximin means selecting the design that scores the best on its lowest scoring criterion, Satisficing sets a threshold for each value and allows for many solutions within those boundaries, Respecification requires human judgment and potentially stakeholder involvement to respecify values such that there is no more conflict. Innovation refers to the development of new design options that solve or ease the conflict. For example, *Value dams and flows* (van der Velden & Mörtberg, 2015b) are a method for innovation. Value dams are technical features that are opposed (as they conflict with important values), while value flows are technical features that are broadly supported by stakeholders.

However, these methods make assumptions that we must take into account. Cost-benefit analysis, direct trade-offs and maximin assume that values are commensurable: they can be directly compared on a common (monetary) scale (van de Poel, 2015a). But some values might be incommensurable, and trade-offs may not be morally acceptable. Satisficing, respecification and innovation do not give the optimal solution to the conflict, but fit within the VSD adage of 'progress, not perfection' (Friedman & Hendry, 2019). While SE practices are already familiar with conflict between requirements, the introduction of values into SE has implications for which trade-offs are allowed.

2.2.3. Translating values to requirements

Values such as fairness are abstract notions and have to be made more concrete in a specific setting or context, so moral values are formulated in terms of evaluation criteria—which may then be operationalized. The value hierarchy is a model for this value specification (Kroes & van de Poel, 2015, p.176-177). A value hierarchy is a specification of values into norms—into context-specific design requirements (van de Poel, 2013). See figure 2.2. The relations within the hierarchy are `<norm/requirement>` ***for the sake of*** `<value/norm>`, which represents a number of different relations between two hierarchical elements x for the sake of y: x can be a means to an end, it can be a goal that contributes to the achievement of y, it may enable the achievement of y without contributing directly, or it may remove an obstacle to y (van de Poel, 2013).

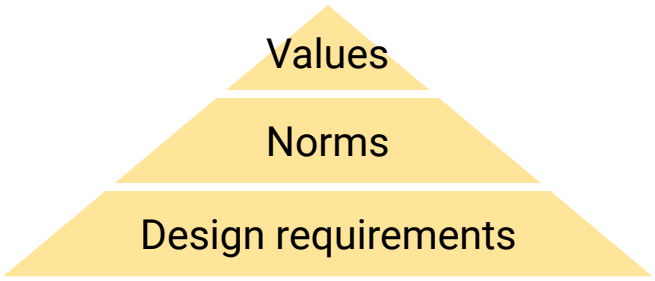


Figure 2.2: The value hierarchy, relationships between levels are not deductive but *for the sake of*

Values—being moral judgments of what is good (or desireable) and what is bad— give rise to norms: "if something is good, it should be pursued" (Miceli & Castelfranchi, 1989, p.181). Norms express duties and obligations (van de Poel, 2013), and which patterns of behaviour are legal or acceptable (Liu, 2000). In other words, properties a technology should exhibit in order to support the value. Technical codes (legal requirements) and standards are an important source for the operationalization and measurement of moral values in design (Kroes & van de Poel, 2015). These documents establish a justified consensus on what constitutes desired behaviour and what should therefore be implemented. There are two criteria for assessing the translation of a value into norm (van de Poel, 2013): 1) Norms should count as an appropriate response to the value, and 2) the norm or set of norms is sufficient to properly respond to or engage with the value.

Design requirements are a further specification how those properties are built in the system. The context-dependent formulation of design requirements by stakeholders makes the value hierarchy a tool

that exposes moral and social value judgments (Aizenberg & van den Hoven, 2020). Requirements can be more specific with respect to (1) the scope of applicability of the norm; (2) goals or aims strived for; and (3) actions or means to achieve these aims (van de Poel, 2013).

Reconstruction of a values hierarchy makes the translation of values into design requirements not only more systematic, it makes the value judgments involved also explicit, debatable and transparent.

Finally, when a value it is important to consider the construct validity and content validity of the value hierarchy conceptualization. Construct validity is the extent to which the norms or requirements are a measurement of the moral value, content validity refers to whether all the relevant aspects of the moral value are taken into account (Kroes & van de Poel, 2015). If this validity is ensured, a values hierarchy is a useful technique to make the translation of values into design requirements systematic, explicates value judgments, and makes these—otherwise implicit assumptions—debatable and transparent.

2.3. Related work

While the proposed Software Engineering for Values (SEfV) framework is an open research question, there are a number of works related to the framework. These are works aimed at combining DfV and AI or software design (section 2.3.1), (section 2.3.1). Works aimed at improving auditability of AI systems (section 2.3.1).

2.3.1. Design software for values

L. P. Simons and Verhagen (2008) combine the main functionalities of VSD and Quality Function Deployment (QFD) to structurally conceptualize values to specify requirements. QFD is a product development methodology that focuses on maximizing customer satisfaction by relating customer needs to product features, using performance measures to score features on their quality (Hauser et al., 2010). The central element is the House of Quality, a matrix relating customer needs to performance measures and products of competitors. QFD is a possible approach for the translation of values into the software design, but it is questionable whether it will maximize customer satisfaction (van de Poel, 2015b). The QFD-VSD approach consists of the following processes: (1) formulating the design problem; (2) stakeholder analysis; (3) their requirements and value impacts; (4) value conceptualization and conflicts; (5) exploring systemic effects; (6) QFD matrix at service concept level; (7) main systemic issues are translated into value scenarios; (8) formulating requirements based on the most pressing issues.

Their approach shows the advantages QFD has over VSD regarding the empirical and technical investigations as it systematically addreses stakeholder priorities, trade-offs, and the relation between priorities and technology choices). However, the proposed design method merely elicit values and prioritize high-level requirements but do not perform the other steps in the SE process.

C. A. Detweiler et al. (2010) formulate six principles for design methods in order to avoid the negative consequences of value violations. They use VSD methods (stakeholder analysis, interviews) to elicit values and propose a method for formalizing values in agent-oriented software engineering requirements in a way that adheres to these principles. These principles are:

1. Values of all stakeholder should be elicited in as far as relevant for the system under design;
2. Stakeholders values should be addressed during all phases of the design process;
3. Conflicts between values of the designers and those of the stakeholders need to be discussed with those who issued the order for the system;
4. To account for the relevant values, the relevant values need to be instantiated explicitly throughout the design process;
5. Design decisions can and need to be justified and evaluated in terms of explicit (instantiations of) stakeholders' values;
6. Conflicts between values need to be made clear and addressed in cooperation with the stakeholders.

Most of these principles focus on the handling of value tensions. By integrating the principles into the objectives of the SEfV framework, we ensure it avoids the negative consequences of value violations.

Value-sensitive Requirements Engineering A number of works have produced requirement engineering methods for values. C. Detweiler and Harbers (2014) introduce a requirement elicitation workshop, which has been validated for usability by Harbers et al. (2015). It comprises of (1) analyzing the system stakeholders, (2) analyzing their values, (3) providing concrete situations, (4) determining the stakeholder needs and (5) creating user stories (value stories). Techniques used are envisioning cards, value dams and flows and brainstorming (see section 2.2). The authors contribute *value stories*, an adaptation of user stories, which are a common requirement elicitation technique (section 2.1.1). Value stories are written in the form of "*As a* `<stakeholder>`*, I want* `<stakeholder need>`*, so that my* `<value>` *is promoted/supported when* `<concrete situation>`". The value stories workshop is an intuitive technique for eliciting requirements, but needs a prompt (like envisioning cards) for effectively analyzing values and concrete scenarios.

D. Simons (2019) developed a workshop for requirement elicitation in design teams. The workshop is useful for value elicitation, but focuses on other aspects than those of our study (software engineering, explication and auditability).

Value-based requirements engineering (Thew & Sutcliffe, 2018) is a requirement elicitation method based on the analysis of stakeholders' values, motivations and emotions.

Esiava (2014) proposes a requirement elicitation and analysis method. The method translates a concrete set of norms to design requirements, formalized with deontic logic, which can be formally checked for consistency and validated with stakeholders. Given a value conceptualized in norms, deontic logic is useful for formulating hard requirements and checking the feasibility of satisfying them all. However, this method does not give room for trade-offs or soft goals.

Participatory software design A number of work use participatory design to include end users their values. WeBuildAI (Lee, Kahng, et al., 2019) and Value Sensitive Algorithm Design (VSAD) (Zhu et al., 2018) provide design methods for the design of ADM. WeBuildAI elicits stakeholders' individual models of fairness, aggregating it into a collective model and providing adequate explanation. VSAD focuses on participatory algorithm selection, collective goals and early feedback.

Conceptual frameworks for value-sensitive software design some works inspect software design at a high-level. Barn et al. (2015) propose a conceptual framework for value-sensitive requirement elicitation, based on participatory design.

Value Sensitive Software Design (Aldewereld et al., 2015) is an approach to represent the design of ICT systems (specifically service-oriented architectures) and explicating the relation between values. However, the authors only provide a theoretical model that integrates business needs, values and the design of IT systems, not methods to guide the software design process.

Research challenges for Software Engineering for Values Mougouei et al. (2018) propose a research roadmap for operationalizing human values in software, identifying the key issues that make it a hard task. They focus on the specification of values and explicating the values of software designers and design practices.

Ethics-Aware Software Engineering (EASE) (Aydemir & Dalpiaz, 2018) is a research roadmap that focuses on the tractability of ethics in software engineering. The EASE development cycle is development process-agnostic, so it is applicable in any software development process. It consists of five stages: 1) Articulation; 2) Specification; 3) Implementation; 4) Verification; 5) Validation.

Brun and Meliou (2018) identify the research challenges within SE, focused directly on the elicitation of fairness requirements.

Holstein et al. (2019) identify research gaps between fair-ML research and practice, based on a survey amongst Machine Learning practitioners.

Aizenberg and van den Hoven (2020) make a case for designing AI for human rights through drawing on the methodologies of Value Sensitive Design and Participatory Design.

Balayn et al. (2020) identify research gaps with respect to biases within database management systems. See appendix **??** for a comparison of research roadmaps.

2.3.2. Improving auditability and transparency of ADM

Research effort has been made to increase auditability and interpretability of AI through data sheets ('nutritional labels' for data). These are the Dataset Nutritional Label (Holland et al., 2018), Datasheets for Datasets (Gebru et al., 2018) and Model cards for model reporting (Mitchell et al., 2019). Stoyanovich (2019), Stoyanovich and Howe (2019) aim to generate labels automatically or semi-automatically by learning. To start this research direction off, they postulate a set of properties a nutritional label should have.

While nutritional labels are useful for data science practices, they are designed to support consumer-specific decisions rather than a complete audit of value judgments behind design decisions (Stoyanovich & Howe, 2019). These studies do provide us with 1) examples for structuring design decisions in an auditable way, 2) a validation of the relevant ADM design parameters (dataset and ML models).

Another method to improve auditability are algorithmic assessments. These are the Algorithmic Impact Assessment framework (Reisman et al., 2018) and the Human Rights, Ethical and Social Impact Assessment (HRESIA; Mantelero, 2018) both are governance model for public agencies, but do not provide operational methods for auditing software systems. Algorithm assessments are therefore not suitable, as our aim is software auditability at an operational level (individual value judgments and design decisions within a software artifact).

3

Conceptualizing fairness in Algorithmic Decision-Making

This chapter conceptualizes fairness

3.1. The concept of fairness

Social systems evolve allocation mechanisms for resources, rights, responsibilities, costs, and burdens (Cook & Hegtvedt, 1983). A central question is how allocation of resources can be fair and just for the individual and society (Colquitt, 2001). Every discipline has their own viewpoints on fairness. For instance, in economics, fairness is achieved through maximizing utilities or minimizing inequality (Sen & Foster, 1997). In sociology, research on fairness investigates which factors contribute to the human fairness judgment within social settings such as families, organizations (and thus socio-technical systems) (Lind & Tyler, 1988).

The research area of organizational justice (OJ) tries to explain how justice impacts the effective functioning of organizations (Colquitt, 2001). Here, fairness is treated as the human judgment of some outcome, interaction or procedure within the organizational context, such as a managerial decision. Since Algorithmic Decision-Making (ADM) is a similar scenario to a managerial decision, namely a decision made by another person (or system) in a hierarchical relationship, OJ is a suitable field for describing the fairness of decision-making.

Within OJ, theories attempt to explain fairness judgment of actions, of which we use the fairness theory of Colquitt et al. (2001) as it provides a validated set of factors that contribute to fairness. There are numerous other theories that attempt to describe fairness, such as the relational models of procedural justice (Tyler & Lind, 1992), fairness theory (Folger & Cropanzano, 1998, 2001) or the moral virtues model (Cropanzano et al., 2001). Fairness theory (Folger & Cropanzano, 2001), for example, integrates earlier theoretical frameworks, with as main goal explaining how people judge accountability (assigning blame or credit) for negative events. However, Colquitt (2001), Colquitt et al. (2001) put effort in synthesizing earlier measures of the concepts that make up the four-dimensional model of fairness, reducing inconsistencies and overlapping factors. Another reason why we prefer it is its adaptability. Colquitt and Rodell (2015) argues that the resulting factors in the four-dimensional model are 'justice rules' that have been applied to different contexts over the years and that the existing rules can be tailored to the context in question. This means we can adapt the factors to our ADM setting.

There are four dimensions of fairness (Colquitt, 2001): (1) *Distributive fairness*, regarding the allocation of outcomes according to a distributive rule; (2) *Procedural fairness* refers to the decision-making process leading up to the decision; (3) *Informational fairness* refers to the justification of the decision and decision-making procedure; and (4) *Interactional fairness* concerns the interpersonal treatment (R. Bies & Moag, 1986). The distinction between the four dimensions of fairness are clarified in table 3.1.

3.1.1. Distributive fairness

Early research focused on fair exchange (with *equity* at the center) and fair decision outcomes. Social scientists established the concept of distributive fairness (Adams, 1965; Deutsch, 1975; Homans, 1961;

Table 3.1: The four dimensions of fairness in sociology

Dimension	Core question
Distributive fairness	Did they get what they deserve or need?
Procedural fairness	Was the decision made in a fair way?
Informational fairness	How well was the decision communicated?
Interpersonal fairness	How were they treated during the decision-making?

Leventhal, 1976) and described general principles of fairness (Deutsch, 1975; Eckoff, 1974; Leventhal, 1976) that are used as distributive rules in social systems. Suppose we have an allocator (such as an ADM) that distributes valued rewards, resources, obligations, rights among a number of recipients (Cook & Hegtvedt, 1983). We speak of distributive fairness if the allocation "is consistent with the goals of that particular situation, such as maximizing productivity or improving cooperation" (Leventhal, 1976). This implies that there is a distributive rule for structuring rewards that engenders distributive fairness. For example, equity— where there is an equivalence of outcome/input ratios —represents fair exchange in order to maximize productivity. In an organizational context, it is the first distributive rule that was studied (Adams, 1965; Homans, 1961), which was later expanded to to include equality and needs (Deutsch, 1975).

Eckoff (1974) provides a framework that captures these different rules as general principles of equality: Objective equality, subjective equality, relative equality, rank order equality, and equal opportunity. The principles differ in their assumption of which characteristics matter for the allocation.

Objective equality states that every individual gets just as much. *Subjective equality* (the needs rule) implies equality of outcome while taking into account need and/or desert. *Relative equality* corresponds to the equity rule, equivalence of income/output ratios. *Rank-order equality* differs from relative equity by rewarding on the basis of position or status, but also precedence ("first come, first serve"). *Equality of opportunity* refers a lack of unfair direct discrimination or indirect discrimination, taking into account social factors such as need. Another principle—the doctrine of sufficiency—states that what is morally important, is that people "have enough" Frankfurt (2015).

Thus, from (Eckoff, 1974) we derive the factors *distributive norm* and *relevant characteristics*. See table 3.2 for an overview.

Table 3.2: Distributive principles of equality (Eckoff, 1974) and sufficiency (Frankfurt, 2015)

Principle	Example	Relevant characteristics					
		Need	Fitness	Desert	Status	Position	None
Objective equality	*Every person gets one meal*						x
Subjective equality	*Those who need more, get more*	x		x			
Relative equality (equity)	*The more you work, the more you get*		x	x			
Rank-order equality	*The higher up you are, the more you get*				x	x	
Equality of opportunity	*Everyone is eligible to apply for the job*	x	x				x
Sufficiency	*Every person gets at least one meal*						x

3.1.2. Procedural fairness

In the next decade, social scientists identified a different notion of fairness, namely procedural justice, observing that — despite fair outcomes — the procedure that led to the distribution might be considered unfair. Contributing factors relate to the distribution of control (whether the individual has control over the process or the decision itself) (Thibaut & Walker, 1975) and fair process criteria (Leventhal, 1980). *Process control* refers to providing control over the presentation of evidence and arguments to the decision-maker. Similar to 'due process' in law, it regards to opportunties to voice one's views and present arguments during a procedure. *Decision control* refers to providing influence the actual

outcome itself.

Leventhal formulated criteria for procedural justice (Leventhal, 1980): consistency, bias suppression, accuracy, correctability, representation, and ethicality.

Consistency where the procedure is applied consistently across persons and time. Consistency in the procedure applied across persons means that similar procedures should be applied to all individuals, giving no special treatment. Consistency in the procedure applied across time means that procedures need to be kept stable, at least over the short term.

Bias suppression refers to the prevention of: 1) personal self-interest and non-neutrality; and 2) narrow preconceptions (e.g. dogmatism, generalization).

Accuracy is defined by Leventhal (1980) as "[basing the decision] on as much good information and informed opinion as possible, where information and opinion must be gathered and processed with a minimum of error". The second part pertains to a robust and secure information system and is left out of scope.

Correctability (Leventhal, 1980) conveys that the procedure should provide (formal and informal) opportunities to review, contest and correct decisions made at various points in the process. *Review* refers to the decision-maker providing the individual with the relevant information that lead up to the decision and identification of the accountable entity. *Contest* is the formal opportunity, where the individual can start an appeal procedure to contest the decision or data used for the decision (for the discretion of a human agent). The determinants of contestability are a safe, easy to use, and an impartial appeal procedure.

Representativeness, according to Leventhal (1980) means that "all phases of the allocative process must reflect the basic concerns, values, and outlook of important subgroups in the population of individuals affected by the allocative process".

The ethicality rule states that allocative procedures must be "compatible with prevailing moral and ethical values accepted by the individual" (Leventhal, 1980). We specify ethicality further into the factors lawfulness and justification. Similar to German et al. (2018) we broaden Leventhal's definition to refer to both the moral standards of the individual and the organization that operates the ADM, as individuals and organizations both have values (Finegan, 2000).

Having broadened the notion of ethicality, 'compatibility with the organizational values' now reads as the extent to which the decision-making procedure is justifiable with the organizational values. Organizational values are "evaluative standards relating to work or the work environment by which individuals discern what is right or assess the importance of preferences" (Dose, 1997).

3.1.3. Interactional fairness

Another factor, identified in the same decade as procedural fairness, was the interpersonal treatment people receive as decision-making procedures are enacted (R. Bies & Moag, 1986). This 3-dimensional specification of fairness is also seen in the fairness triangle, developed by multiple Ombudsmen[1], consisting of substantive – procedural – relational fairness.

Greenberg (1993) researched interactional justice in terms of explanations and sensitivity, proposing a four-factor model where interactional justice has been further subdivided into two different factors: interpersonal and informational justice.

Colquitt uses Greenbergs designations of ***interpersonal fairness*** (sensitivity) and ***informational fairness*** (explanation). Interpersonal justice consists of Bies & Moag's (1986) constructs *respect* and *propriety*. For a further specification of these constructs, we look at R. J. Bies (2015). He refers to them as types of profanities that violate the human dignity, where protection from these profanities is considered as interpersonal fairness. There are three categories: disrespect, invasion of privacy, and exposure to personal dangers. Figure 3.4 gives an overview of the full specification of interpersonal fairness.

Disrespect means actions that demean or devalue the worth of an individual and consists of *inconsiderate actions* (failing to meet people's minimal expectations for considerate treatment, such as untimely feedback, a lack of explanation or account of decisions that affect them) and *abusive actions*. Abusive actions come in many forms: rudeness or impoliteness, public criticism and beratment[2], *ac-*

[1] Ombudsman = a government official […] appointed to receive and investigate complaints made by individuals against abuses or capricious acts of public officials. (from https://www.merriam-webster.com/dictionary/ombudsman)

[2] berating = to scold or condemn vehemently and at length (from https://www.merriam-webster.com/dictionary/berating)

tions intended to embarrass and humiliate a person publicly, and *prejudicial statements* (usually racist or sexist).

Invasion of privacy may be in the form of *disclosure of confidences and secrets* private matters that are supposed to be held in confidence but also the disclosure of personal information without their permission. Another form is *intrusive information-gathering* such as the asking of improper questions (e.g. marital status, whether someone wants to have children), information that is not relevant for the context, or a information-gathering procedure that is too psychologically intrusive or invasive.

Finally, *exposure to personal dangers* refers to violations of an individual's psychological safety, mental health and physical safety. *Psychological safety* refers to both mental health (psychological pain and physiological stress (R. J. Bies, 2015) and "feeling able to show and employ one's self without fear of negative consequences to self-image, status, or career." It is influenced by elements of social systems that create situations that are predictable and consistent. (Kahn, 1990) With *physical safety*, R. J. Bies (2015) refers to both physical consequences of stress (e.g. not being able to sleep) as well as direct dangers due to poor working conditions.

Interpersonal fairness support Compared to Colquitt (2001), the EU Charter of Fundamental Rights (ECFR;European Union (2016a)) provides much broader guidelines about exposing individuals to dangers, namely the protection of physical integrity. Moreover, since the violation of an individual's psychological and physical safety "raises the interactional [fairness] issues of human rights and human dignity" (R. J. Bies & Greenberg, 2017), a specification on the basis of internationally recognized human rights is justified.

Physical integrity refers to the inviolability of the physical body in the sense of physical harm physical and self-determination or autonomy (Child Rights International Network, n.d.).

According to Bublitz (2013), mental integrity (Art. 3 ECFR) may be interpreted as both a right to mental health, and a right to cognitive liberty: the right to mental self-determination, guaranteeing an individual's sovereignty over their mind. The right to mental integrity defends an inner sphere of liberty (within the mind) (Craig, 2016, p.1). Note that that Colquitt's notion of psychological and physical safety are implied by these definitions.

Informational fairness items The **informational fairness** dimension is is based on Bies & Moag's 1986 other two constructs — truthfulness and justification — where the latter is specified with the factors that improve the perceived adequacy of explanations: *reasonableness*, *timeliness*, *specificity* (Shapiro et al., 1994). The former is specified into *deception*, *derogatory judgments* and *transparency* about the decision-making process. Besides transparency, these factors need to be further specified (figure 3.3 gives an overview of the full specification of informational fairness). *Specificity* is the extent to which the explanation is personalised, specific and not generic or 'canned'. *Timeliness* refers to the amount of time the individual has to wait for the explanation of the decision. *Reasonableness* How reasonable the explanation is. In other words, whether the explanation allows individuals to understand the causes of the (negative) decision (Shapiro et al., 1994, p.349).

R. J. Bies (2015) specifies two violations of truthfulness in the same way that violations of human dignity make up interpersonal fairness: deception and derogatory [3] judgments. *Deception* is the "lack of correspondence between one's words and actions" (R. J. Bies, 2015), and may take place in the form of bluffing, misrepresentation of position, deception or falsification. Note that the justification of deception depends on the context, as according to Anton (1990) in negotiation, people do not necessarily see certain types of deception as unjust, but as a bargaining strategy.

Derogatory judgments are a "lack of truthfulness and accuracy of statements and judgments about a person" (R. J. Bies & Tripp, 1993), legally known as libel or slander. Wrongful or unfair accusations (such as accusing an individual of stealing ideas), managers blaming people for failures of their own responsibility, or creating an unfavorable image of a person.

Adapting the conceptualization of Colquitt (2001) gives us a specification of *fairness* into four dimensions, specified into factors that influence human judgment over decision-making that is similar to ADM. This is not a definitive list of criteria, but proven to be concepts that are important to their aspects of fairness judgments. Colquitt (2001) points out that these concepts overlap with many fairness concepts found in other work, but that his conceptualization establishes a cohesive set of criteria that independently measure the abstract notions of distributive, procedural and interactional fairness.

[3] derogatory = diminishing the importance, value or effectiveness of the character or standing of the subject (from https://www.merriam-webster.com/dictionary/derogatory)

As Colquitt and Rodell (2015) argue, these factors have been well-established in sociological research, such that they generalize to 'justice rules' that can be adapted to specific contexts. This means that we can step from the descriptive domain into the normative domain and use these justice rules as foundation of the holistic conception of fairness in ADM systems. Table 3.3 shows all four dimensions and these justice rules.

Table 3.3: Organizational justice rules (Colquitt & Rodell, 2015)

Type	Name	Description
Procedural	Process Control	Procedures provide opportunities for voice
	Decision Control	Procedures provide influence over outcomes
	Consistency	Procedures are consistent across persons and time
	Bias Suppression	Procedures are neutral and unbiased
	Accuracy	Procedures are based on accurate information
	Correctability	Procedures offer opportunities for appeals of outcomes
	Representativeness	Procedures take into account concerns of subgroups
	Ethicality	Procedures uphold standards of morality
Distributive	Equity	Outcomes are allocated according to contributions
	Equality	Outcomes are allocated equally
	Need	Outcomes are allocated according to need
Interpersonal	Respect	Enactment of procedures are sincere and polite
	Propriety	Enactment of procedures refrain from improper remarks
Informational	Truthfulness	Explanations about procedures are honest
	Justification	Explanations about procedures are thorough
	Derogatory judgments	Explanations about procedures are non-derogatory

3.2. Fairness in ADM

In this section, we adapt the conceptualization of fairness to our ADM context, further specifying the factors at a lower level (e.g. data quality). For distributive and informational fairness, we do not further need to specify to the context of ADM.

Distributive fairness in ADM The specification from section 3.1.1 is adequate. See figure 3.1 for a specification of distributional fairness for ADM.

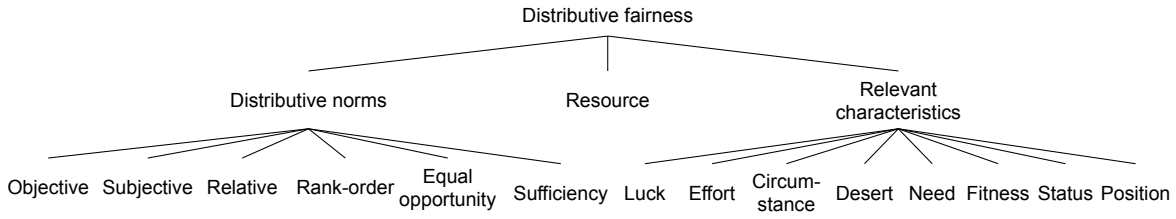


Figure 3.1: Specification of the distributive fairness dimension for ADM

Procedural fairness in ADM Outcome bias recognition and mitigation: which biases are undesired (racist, sexist, ageist, etc.). Desired biases (such as granting less risky loans to people who have less

steady income) are considered to be distributive fairness. As found in chapter 4, bias suppression is the main focus of the computer science community.

Accuracy of information refers—in an ADM context—to the quality of the data and the performance of its predictive algorithms/models.

Depending on the organization, context-specific data quality dimensions are preferred. However, for further specification, we use Wand and Wang (1996) their ontology of data quality dimensions for information systems. We can make a distinction between an external view of the ADM (i.e. the use and justification of the 'black box' information system for the real world) and an internal view (the construction and operation of the ADM to achieve its function in the real world).

External Data Quality is concerned with the how the ADM as a 'black box' is usable and valuable in the real world. The most common dimensions are timeliness, relevance, content, importance, sufficiency, usableness, usefulness, clarity, conciseness, informativeness, level of detail, freedom from bias, quantitativeness, scope, interpretability, understandability.

Internal Data Quality dimensions reflect the extent to which the data represents the real world. Common dimensions are accuracy, reliability, timeliness, completeness, currency, consistency, precision, reliability. However, as the authors note, the definitions are ambiguous.

The *model performance* of the ML model describes the quality of the ADM's inferred information on the individual (e.g. whether a person is high risk). Depending on the context and model type the performance metric given as classification accuracy, recall, f-score, area under curve, mean absolute error, mean squared error, etc. This means that the selection of evaluation/benchmarking criteria is also an aspect of procedural fairness.

Few ADM are designed to easily allow individuals to contest adverse or incorrect decisions (Whittaker et al., 2018, p.19)

Contestability and human intervention in ADM has received attention since modern data protection laws such as GDPR have included a right to human intervention (GDPR Art. 22).

Almada (2019) suggests a possibility for designing for contestability would to ensure that data subjects will have the necessary information and tools to exercise the right to intervention; and that ADM provide adequate interfaces to allow contestability.

Under correctability, we place the factor *human oversight & correction*, representing the extent of human control over the ADM, i.e. the extent to which the decision can be corrected by a human operator. The act of correcting might be a complex and tedious process within a bureaucratic organization (which is perceived as less fair), and is therefore not a trivial aspect of correctability. On top of that, in the context of ADM, the correction should propagate back into the ADM system (retraining) to remain consistent.

Human oversight can be achieved through mechanisms such as *human-in-the-loop* (capability for human intervention in every decision cycle), *human-on-the-loop* (human intervention during the design cycle and the system's operation), *human-in-command* (capability to oversee the overall activity of the AI system) (High-Level Expert Group on Artificial Intelligence, 2019).

Colquitt (2001) decided to leave out the representation criterion, because it "subsumes process control [and] decision control" (p. 3), however in the setting of software development there is a distinction between representation during the software design & development cycle and the individual decision-making process.

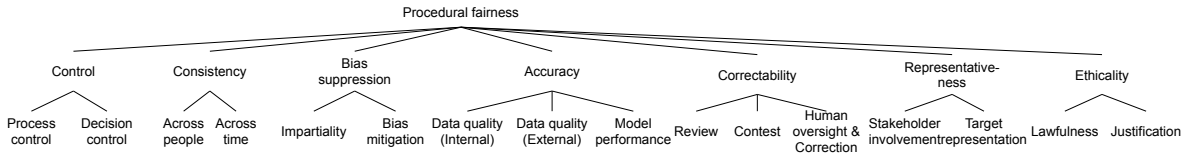


Figure 3.2: Specification of the procedural fairness dimension for ADM

Informational fairness in ADM The specification from section 3.1.3 is adequate. See figure 3.3 for a specification of the informational fairness dimension.

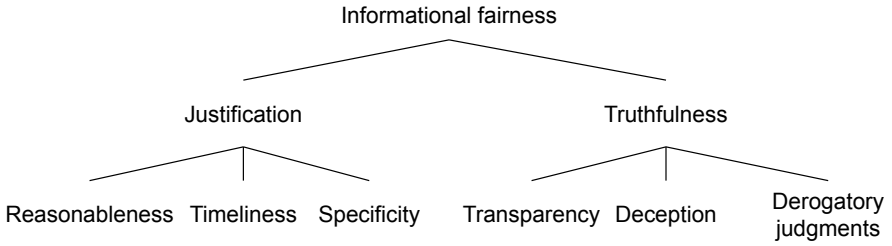


Figure 3.3: Specification of the informational fairness dimension for ADM

Interactional fairness in ADM Adapting for ADM seems unnecessary. While the factors that make up *disrespect* seem only human behaviours, ADM are capable of disrespecting human beings too, varying from inconsiderate behaviour to perpetuating stereotypes in prejudicial statements. For instance, a recent example of online ad delivery system associating black-identifying first names (DeShawn, Darnell, Jermaine) more with the word "arrest" (Sweeney, 2013). An ADM system could also make abusive or prejudicial statements, such as a search engine's autocomplete giving offensive prompts [4].

Privacy-related issues arise with data-driven systems. An important privacy aspect overseen by Colquitt (2001) is the unjustified **use** of an individual's personal data and automated decision-making based on personal data (European Union, 2016b, Art. 22).

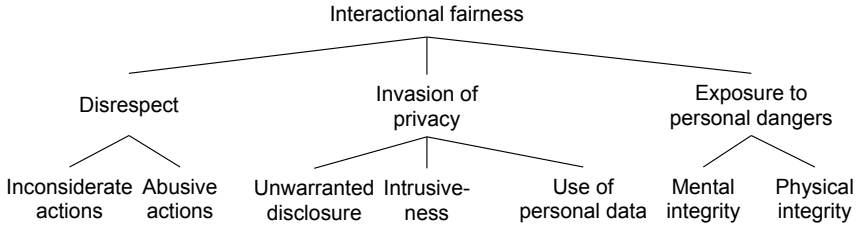


Figure 3.4: Specification of the interpersonal fairness dimension for ADM

In conclusion, table 3.4 shows our fairness specification for the context of ADM systems, based on Colquitt (2001), supported by various sources and adapted to the Computer Science domain.

[4] https://www.wired.com/story/google-autocomplete-vile-suggestions/ (accessed 21-7-2020)

Table 3.4: Our conceptualization of the value fairness in automated decision-making systems

Fairness dimension	Concept	Specification	Description
Procedural	Control	Process control	The ADM shall provide the individual sufficient control over the procedure.
		Decision control	The ADM shall provide the individual sufficient influence over the decision outcome.
	Consistency	Across individuals	The ADM shall apply decision-making procedures consistently across individuals.
		Across time	The ADM shall apply decision-making procedures consistently across time.
	Bias Suppression	Impartiality	The ADM shall be neutral and guard against those with an interest in the decision.
		Bias suppression	The ADM shall suppress undesireable outcome biases.
	Accuracy	Internal data quality	The ADM shall use data that sufficiently represents the real world.
		External data quality	The ADM shall use data that is sufficiently usable and valuable for its intended functionality
		Model performance	The ADM shall use predictive models with high performance.
	Correctability	Review	The ADM shall easily allow individuals to review the relevant information leading up to the decision and the accountable entity.
		Contest	The ADM shall easily allow individuals to contest adverse or incorrect decisions.
		Human oversight & correction	The ADM shall allow human operators to oversee decision-making and correct decisions.
	Representativeness	Subgroup involvement	All phases of the ADM shall involve the important subgroups in the population of individuals affected by the ADM.
		Target representation	The ADM shall aim for a certain representation of subgroups in the target population.
	Ethicality	Lawfulness	The ADM shall operate in accordance with applicable laws and regulations.
		Justification	The goal and practices of the ADM shall be justifiable within the organizational and societal values.
Distributive	Distributive Norms		The ADM shall allocate the resources (outcomes) in a manner consistent with its goals.
	Characteristics		The ADM shall take specific relevant characteristics into account.
Informational	Justification	Reasonableness	The ADM shall provide a reasonable explanation.
		Timing	The ADM shall provide a justification within a reasonable amount of time.
		Specificity	The ADM shall provide a sufficiently personalised and specific explanation.
	Truthfulness	Transparency	The ADM shall have a transparent decision-making process.
		Deception	The ADM shall have correspondence between its claims and actions.
		Derogatory Judgments	The ADM shall not make statements or judgments that lack truthfulness or accuracy.
Interpersonal	Disrespect	Inconsiderate Actions	The ADM shall meet people's minimal expectations for considerate treatment.
		Abusive actions	The ADM shall not perform harsh, insulting language or actions.
	Privacy	Unwarranted disclosure	The ADM shall not unwarrantedly disclose personal information, secrets and confidences.
		Intrusiveness	The ADM shall not perform intrusive information-gathering.
		Use of personal information	The ADM shall not unjustifiably use personal information.
	Exposure	Mental Integrity	The ADM shall respect the individual's right to mental integrity.
		Physical Integrity	The ADM shall respect the individual's right to physical integrity.

3.3. Discussion

The first contribution of this study is a specification of fairness into high-level norms that should be considered for any ADM design. Organizational justice provides us with a multidimensional specification of fairness that applies specifically to decision-making within the organizational context. The organizational justice conceptualization has been used by other authors as well (Binns et al., 2018; German et al., 2018; Lee, Jain, et al., 2019; Robert et al., 2020).

We must realise, however, that the fairness theory from organizational justice is a descriptive, behavioral theory. On one hand we can argue that such a theory is desirable, since fairness in ADM is inherently a prevailing judgment of the perceived behaviour of a machine. On the other hand, we must not mistake it for normative ethical theory, despite them having become 'justice rules' (see table 3.3) in the same way nature has 'laws'. This means that we should consider the conceptualization as descriptive and not normative (i.e. it should be suggestive and elicitive), and it should be extended/supported by other domains such as law, philosophy. For instance, depending on the context a norm such as *decision control* (where the subject should have the ability to disregard or change the outcome) might not be applicable in some cases (e.g. in recidivism risk prediction).

4

Current practices in designing for fairness

This chapter contains a literature review for existing fairness definitions and their corresponding software design practices, in order to answer research question 2.

The chapter is structured as follows: We clarify the terminology around fairness in section 4.1. Fairness measures and interventions are grouped by their assumption of fairness: fairness as non-discrimination (section 4.2), fairness as fair representation (section 4.3) and fairness considerations beyond allocation and representation (section 4.4). In section 4.5 the discovered practices are related to the fairness conceptualization, and finally we discuss our findings in this chapter (section 4.6).

4.1. Fairness terminology in CS

Different terms are interrelated in machine learning (ML) and data mining practice: bias, discrimination, equality and fairness. Here we aim to create some clarity. Most studies consider fairness to be non-discrimination: an ”absence of any prejudice or favoritism toward an individual or a group based on their inherent or acquired characteristics” (Mehrabi et al., 2019). This non-discrimination is implicitly achieved by ensuring equality of the resource (e.g. granting loans) across these groups and individuals.

Non-discrimination in the context of machine learning (Žliobaitė, 2017) is defined as:

1. (Direct discrimination) People that are similar in terms of non-protected characteristics should receive similar predictions;

2. (Indirect discrimination) Differences in predictions across groups of people can only be as large as justified by their non-protected characteristics.

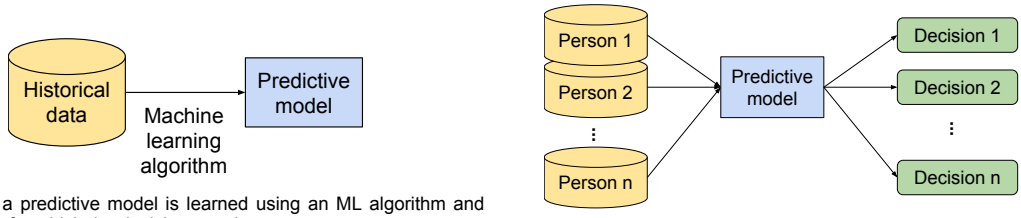
Bias typically refers to two different phenomena: data bias and outcome bias. *Data bias* is a systematic distortion in the data (Olteanu et al., 2019) and may cause unfair and inaccurate predictions if left untreated (Barocas & Selbst, 2016).

Outcome bias of a ML system is the observed statistical preference in the aggregated decisions of a deployed predictive model (figure 4.1b). Formally, a statistical estimator is biased if its expected or average value differs from the true value that it aims to estimate (Barocas, Hardt, et al., 2017). We would see that the algorithm seems to favour persons with a certain attribute.

For outcome biases, it is useful to distinguish between undesired, desired and unimportant bias (Balayn et al., 2020). A *desired bias* is planned and not considered unfair, i.e. an (implicit) distributive rule (see section **??**). An example of desired bias is a credit risk system that is biased against individuals with low income. This would correspond to the distributive fairness norm of ’equity’, where those who contribute more, get more.

Unimportant biases are unplanned for nor are they experienced as negative.

A bias is undesired if it is considered unfair by the stakeholders of the system, usually on the basis of protected characteristics (race, sex, religion, age, etc.). Thus, undesired bias corresponds to indirect discrimination (e.g. a credit risk system that is biased against women).



(a) Training: a predictive model is learned using an ML algorithm and historical data for which the decisions are known

(b) Operation: the predictive model makes a decision based on the features of the new person

Figure 4.1: A typical machine learning setting during training (a) and operation (b)

4.2. Fairness as non-discrimination

A majority of the work defines fairness as a mathematical condition for non-discrimination with a corresponding intervention to enforce the condition. We aim not to enforce a definition, but to categorize them. Within practices for ensuring non-discrimination, the following categories of fairness exist: fairness through unawareness, statistical measures, similarity-based and causal reasoning (Verma & Rubin, 2018).

4.2.1. Unawareness

Fairness through unawareness (FTU) is achieved when the protected attributes are not used in the decision-making process (Kusner et al., 2017). This notion of fairness avoids direct discrimination, but not indirect discrimination as other features may serve as a proxy for the protected attribute (Zliobaite, 2015).

4.2.2. Statistical measures

Statistical measures are based on parity of outcome or error between groups (e.g. gender, age). The confusion matrix (table 4.1) is a helpful tool for visualizing. A confusion matrix (also called 'contingency table') shows the performance of the algorithm in terms of the number of (in)correctly classified individuals. True positives (TP) are those who are correctly classified in the positive outcome class, true negatives (TN) for the negative outcome class. False positives (FP) are those who deserved a negative outcome (e.g. not financially stable enough for a loan), but were classified in the positive outcome class. False negatives (FN) deserved a positive outcome but were given a negative outcome. Performance measures are based on these four counts. For instance, *precision* is the positive predictive value, calculated as $PPV = \frac{TP}{TP+FP}$ Statistically, precision corresponds to the probability that a person who is classified in the positive outcome class, to actually belong in that class ($P(Y = 1|\hat{Y} = 1)$). Statistical measures of fairness require the confusion matrix-based performance measures (e.g. precision, recall, accuracy) to be equal across groups.

male	$y = 1$	$y = 0$	total
$\hat{y} = 1$	600 (TP)	200 (FP)	800
$\hat{y} = 0$	400 (FN)	300 (TN)	700

female	$y = 1$	$y = 0$	total
$\hat{y} = 1$	300 (TP)	200 (FP)	500
$\hat{y} = 0$	200 (FN)	300 (TN)	500

Table 4.1: Confusion matrices examples for men (left) and women (right). y is the true label, $\hat{y}$ is the decision (c.f. Berk et al. (2018))

There are three statistical criteria of non-discrimination (Loi et al., 2019): Independence, separation and sufficiency (see figure 4.2). Independence relates to only the decisions $\hat{Y}$ (third column), separation compares outcomes vertically (e.g. TP/TP+FN), sufficiency compares outcomes horizontally (e.g. TP/TP+FP).

In binary classification they mean the following:

Independence ($Y \perp G$): Individuals have the same statistical prospects of being of either true label, regardless of their group membership. This family of statistical measures consists of statistical parity, conditional statistical parity, normalized difference (Dunkelau & Leuschel, 2020).

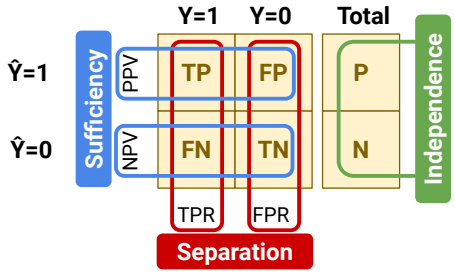


Figure 4.2: Confusion matrix and its relation to the three statistical criteria of non-discrimination: sufficiency, separation and independence. $\hat{Y}$ is the predicted score, Y the true label

Statistical parity holds if both groups have the same probability of the positive outcome, so $P_1(\hat{y} = 1) = P_0(\hat{y} = 1)$.

A relaxation of statistical parity exists (ϵ-fairness) to allow for some disparity. If $\epsilon = 0.2$, it corresponds to the 80%-rule in US employment law: There may only be a 20% difference between outcome for groups (Dunkelau & Leuschel, 2020). In our example, $P_m(\hat{y} = 1) - P_f(\hat{y} = 1) = 0.67 - 0.5 = 0.17$, so the classifier would be 20-fair.

Conditional statistical parity is an adaptation of the former, where statistical parity should hold for a limited set $L \subset X$. For example, defendants who have the same number of prior convictions (L) should have equal detainment rates (Corbett-Davies et al., 2017).

Normalised difference gives the magnitude of statistical parity between 0 and 1. (Žliobaitė, 2017)

Sufficiency ($\hat{Y} \perp G|Y$): "individuals with the same true label have the same statistical prospects of either decision, regardless of their group membership"

Sufficiency is a conditional independence, by taking the ground truth Y of the samples into account. In the confusion matrix it would mean parity of Positive Predictive Value (PPV or *precision*) and/or Negative Predictive Value (NPV). *Predictive parity* requires parity of the probability that a positive classified sample is a true positive. $P_1(y = 1|\hat{y} = 1) = P_0(y = 1|\hat{y} = 1)$.

NPV parity requires parity of the probability that a negative classified sample is a true negative $P_1(y = 0|\hat{y} = 0) = P_0(y = 0|\hat{y} = 0)$.

Conditional use accuracy equality is the conjunction of predictive parity and negative predictive value, and thus a stricter form of sufficiency. Our example does not satisfy this fairness definition since it does have equal PPV ($P_m(y = 1|\hat{y} = 1) = P_f(y = 1|\hat{y} = 1) = 0.6 - 0.6 = 0$) but a difference in NPV $P_m(y = 0|\hat{y} = 0) = P_f(y = 0|\hat{y} = 0) = 0.428 - 0.6 = 0.172$.

Calibration is a fine-grained version of the above notions of non-discrimination (predictive parity), where probabilistic classifiers calculate a risk score, and based on a threshold will classify accordingly. A predicted probability score is *calibrated* if for any score s, subjects in both groups have equal probability to truly be of the positive class. Formally: $P_1(y = 1|S = s) = P_0(y = 1|S = s)$ for $s \in [0, 1]$.

Well-calibration requires the probabilities not only to be equal, but also equal to the score s.

Separation ($Y \perp G|\hat{Y}$): Individuals about whom the same decision is made have the same statistical prospects of being of either true label, regardless of their group membership.

Similar to sufficiency, separation takes the ground truth into account. True Positive Rate (TPR or *recall*) and False Positive Rate (FPR or *selectivity*).

Predictive equality requires FPR parity: $P_1(\hat{y} = 0|y = 0) = P_0(\hat{y} = 0|y = 0)$.

Equality of Opportunity—the complement of predictive equality—requires parity of FNR or TPR. Formally, equal TPR means $P_1(\hat{y} = 1|y = 1) = P_0(\hat{y} = 1|y = 1)$.

Equalised Odds requires the classifier to satisfy both predictive equality and Equality of opportunity.

Separation also contains two fairness definitions based on predicted probability scores instead of classifications. *Balance for the positive class* holds if the average score S of the subject in the positive class is equal for both groups: $E_1(S|Y = 1) = E_2(S|Y = 1)$. Closely related to equal opportunity.

Balance for the negative class is similar, but for subjects in the negative class, so: $E_1(S|Y = 1) = E_2(S|Y = 1)$. This is closely related to predictive equality.

Other statistical notions *Overall accuracy equality* requires both groups to have an equal accuracy, so $P_1(\hat{y} = y) = P_0(\hat{y} = y)$. Does not distinguish between accuracy for successes and for failures, so in many settings it requires a cost-weighted approach (Berk et al., 2018).

Treatment equality (Berk et al., 2018) if the ratio between false positives and false negatives is equal among groups.

Fairness Under Unawareness (Chen et al., 2019) is a design practice for ensuring notions statistical parity when the protected attribute is not available.

4.2.3. Individual fairness

Individual fairness considers an algorithm to be fair if it gives similar predictions to similar individuals. Existing measures for individual fairness differ in how strict this rule should apply.

Causal discrimination is a strict version of individual fairness and requires that there are no two individuals with equal attributes that are treated dissimilarly (Galhotra et al., 2017).

Fairness through awareness: The distance between predicted outcomes must not be greater than the distance between individuals (Dwork et al., 2012). Formally, a classifier satisfies individual fairness if it satisfies the Lipschitz property: for every two individuals x, y, we have $D(M_x, M_y) \leq d(x, y)$. This notion of fairness requires a distance metric for individuals. Lahoti et al. (2019) proposes a method based on pairwise comparison rather than the calculation of a distance metric.

Other notions of individual fairness include value judgments about the relevant characteristics of the individuals. Joseph et al. (2016) takes into account the individuals' qualifications (whether they meet the required need for the positive classification), in essence meaning *"Less qualified individuals should not be favored over more qualified individuals"*.

Speicher et al. (2018) take desert into account (whether they deserve the treatment), meaning *"Individuals deserving a similar treatment, receive similar treatment"*. This fairness notion is based on economic theory (section 4.2.4).

4.2.4. Measures based on economic theory

Another development in the field of Fair Machine Learning is the application of economic theories of distributive justice (Gummadi & Heidari, 2019). Economic models define individual costs or benefits of being assigned to the positive class. Existing notions of group fairness, predictive parity and individual fairness are proven to be special cases of these economic models (Heidari et al., 2019). There are four different types of fairness definitions based on economic theories: equality of opportunity, (in)equality, social choice theory and fair division.

Equality of Opportunity Heidari et al. (2019) propose a set of measures based on economic models of equality of opportunity (EOP), and show that it unifies most parity definitions, statistical parity, equality of odds, equality of accuracy, predictive value parity). The EOP framework requires the modeler to specify which features are considered *effort* (the subject is considered morally accountable) and *circumstance* (morally not accountable).

Heidari et al. (2019) cast a new light on the impossibility theorem: The moral assumptions about the effort-based utility of individuals are irreconcilable, hence it is impossible to optimize for multiple statistical fairness definitions.

Yaghini et al. (2019) take a descriptive ethics approach in order to allow practitioners to estimate the parameters of the EOP model (Heidari et al., 2019) through human judgment.

Measurement of Inequality The concept of distributional fairness relates to some form of equality. Given a distribution of *benefits*, inequality indices capture the extent to which it is unequal.

Speicher et al. (2018) provides an approach to use inequality indices for measuring individual and group unfairness. Their aim is to measure the extent to which the outcomes of an algorithm unequally benefit different individuals or groups in a population.

Deldjoo et al. (2019) use a similar approach for fairness—Generalized Cross Entropy—, however it allows the designer to specify a distribution that is considered fair—unlike Speicher et al. (2018), who enforces equality of the distribution. Deldjoo et al. (2019) is unique (within our survey) in its assumption that fairness is not necessarily equality between groups, but a proper distribution of utility (benefits).

Instead of individual benefits, a risk-based approach has also been put forward by Williamson and Menon (2019). They propose the conditional value at risk (CVaR) measure for subgroup risk. The CVaR function has a tuning parameter $\alpha \in [0, 1]$ where $\alpha \to 1$ corresponds to the maximin principle of Rawls (section **??**) and $\alpha \to 0$ corresponds to the minimization of the average subgroup risk.

Social Choice Theory Social choice theory (SCT) deals with the fair aggregation of individual preferences (utilities). If these utilities are comparable, they can be aggregated via a cardinal social welfare function.

Heidari et al. (2018) propose a family of measures that correspond to cardinal social welfare, justified by the fairness concept of the "veil of ignorance" (Rawls, section **??**). They show that achieving high social welfare generally implies low inequality in practice, and argue that their measures are easier to integrate in the machine learning pipeline than Speicher et al., 2018.

Ben-Porat et al. (2019) propose a welfare-equalizing fairness constraint under which the disadvantaged group is better off, despite a selfish decision-maker.

Fair division Different entities may have different preferences with respect to the distributed resource. The fairness of a division depends on envy-freeness and equity. Envy-freeness means that no individual has a higher utility for another individuals outcome (i.e. envies another outcome). Equitability means that all individuals have the same utility for their own outcome.

Zafar (2019) introduces two definitions of fairness based on envy-freeness, arguing that parity is too stringent. These are relaxation of disparate treatment and disparate impact, as protected groups do not need equal treatment, but only to prefer their treatment over the treatment they would have received if they were in another group.

Hossain et al. (2019) propose two measures for designing *group envy-free* or *group equitable* classifiers. Benefits of this approach is that it is possible to arbitrarily define groups–meaning that it also optimizes for subgroups; it works for both Y being a ground truth (loan setting) or not (targeted ad setting); and it allows to make trade-offs between loss and fairness.

4.2.5. Causal reasoning

Whereas the previous fairness definitions consider the outcome and treatment, we can also reason about the attributes and their relation to the target variable—before the classifier is built. Causal reasoning assumes a graph that represents attributes (nodes) and direct causal relations (edges). Causal definitions of fairness are based on the existence of a path from the protected attribute G to the predicted variable $\hat{Y}$.

Counterfactual fairness holds if there is no path from G to $\hat{Y}$ (Kusner et al., 2017). In essence, it compares the same individual with a different version of themselves.

No unresolved discrimination if there is no path G to $\hat{Y}$, except via a resolving attribute. A resolving attribute is dependent on G but in a matter which is accepted as non-discriminatory.

No potential proxy discrimination—similar to unresolved discrimination—holds if there is no path from G to $\hat{Y}$ that is blocked by a proxy variable.

No proxy discrimination holds if—after canceling the influence of proxy P on the other attributes X—there is no potential proxy discrimination.

Fair inference if there are no illegitimate paths from G to $\hat{Y}$—where legitimate paths are selected specific to the domain.

4.2.6. Selecting a fairness measure

The following guidelines and toolkits have been researched in order to help with selecting a fairness measure from table 4.2. Fairness through unawareness is legally necessary, but only resolves direct discrimination (Dunkelau & Leuschel, 2020). Statistical independence is important, but not sufficient (Žliobaitė, 2017). Normalised difference gives the degree of statistical independence (1 = fully discriminating, 0 = no discrimination).

Fairness criteria using the true label y, implicitly assume that the true labels are objective (i.e. no discrimination in the historical data). Depending on the context, this assumption is false (Žliobaitė, 2017).

Total fairness—where all statistical fairness conditions are satisfied—is shown to be impossible (Berk et al., 2018). More specifically, statistical parity, equalised odds and conditional use accuracy equality are mutually exclusive (Dunkelau & Leuschel, 2020).

The impossibility theorem shows that *sufficiency* and *separation* cannot be satisfied together, except in rare circumstances. Given that failure of sufficiency is an instance of indirect discrimination, failure of separation is an instance of cognitive discrimination, Loi et al. (2019) provide guidance in choosing between these two fairness norms:

1. If the decision $\hat{Y}$ is the source of benefits/harms to the individual, and the true label Y justifies inequality ⇔ the rule should satisfy separation
2. If the true label Y is the source of benefits/harms to the individual, and the decision $\hat{Y}$ justifies inequality ⇔ the rule should satisfy sufficiency

Table 4.2: Families of ML fairness measures (G: protected attribute, X: other attributes, $\hat{Y}$: prediction, $S_{\hat{Y}}$: predicted probability score, Y: target variable, U: utility, B: benefit)

Category	Factors	Definitions
Unawareness	-	FTU
Statistical independence	$G, \hat{Y}$	SP, Conditional SP, Normalised difference
Statistical sufficiency	$G, \hat{Y}, S_{\hat{Y}}, Y$	Predictive parity, NPV parity, calibration
Statistical separation	G, Y	Equalized Odds, Predictive equality, balance for positive class
Individual fairness	$G, X, \hat{Y}, Y$	FTA, PFR+
Economics measures	$G, \hat{Y}, U, B$	EOP, Inequality indices, social welfare, fair division, CVaR
Causal reasoning	$G, X, \hat{Y}$	CF, NUD, No Proxy, Fair inference

Toolkits While there is a number of tools for aiding developers in the exploration of biases and implementation of bias mitigation techniques, few help developers choose fairness measures. Toolkits for exploring bias include AI Fairness 360[1], What-if tool[2], FAT forensics[3], audit-ai[4].

Two toolkits help in the selection of a suitable fairness measure: The *Aequitas* tool (Saleiro et al., 2018) provides a fairness tree that guides the data scientist and policy maker in finding a suitable measure from: equal selection parity, counterfactual fairness, statistical independence, sufficiency and separation.

Ruf et al. (2020) propose a similar flow diagram for selection between unawareness, individual fairness, statistical parity, equalized odds (Hardt et al., 2016) and its relaxation equalized opportunities.

4.2.7. Techniques

Existing interventions can be classified into the following categories:

1. **Pre-processing:** removing biases from the training data, so the classifier trains on fair examples.
   (a) Relabelling: Changing the ground truths (Y) of the training set to satisfy a fairness definition.
   (b) Resampling: Altering the impact of specific samples during training.
   (c) Fair representations: Mapping features to a debiased feature space.
2. **In-processing:** Modify learning algorithms in order to remove discrimination during the model training process (Mehrabi et al., 2019).

[1] https://aif360.mybluemix.net/
[2] https://pair-code.github.io/what-if-tool/
[3] https://fat-forensics.org/
[4] https://github.com/pymetrics/audit-ai

(a) Adjusted learning algorithms are new algorithms, specifically designed to increase fairness.

(b) Adapted loss functions: The whole loss function is changed or a regularization term is added.

(c) Adversarial approaches: Using adversarial training, where two models play against each other. One model predicts the ground truth, another model predicts the protected attribute based on the first model's prediction.

(d) Optimisation subject to fairness constraints: Instead of changing the loss function, the optimization strategy is subjected to a constraint. The converse is also a possible approach, where we maximize fairness with accuracy constraints.

(e) Compositional approaches, where multiple models are trained for each subgroup.

3. **Post-processing:** A trained classifier is assumed to be biased. The bias is corrected with respect to the protected attribute. Using a holdout set, which is not used during the training of the model. Reassignment function that produces *derived classifier* $\hat{Y}$, optimized to satisfy a fairness definition.

   (a) Output correction

   (b) Input correction: adding preprocessing layers to an already trained algorithm.

   (c) Classifier correction: From a possibly unfair predictor, a predictor is derived which satisfies the fairness definition.

4.3. Fairness as fair representation

Crawford and Calo (2016) distinguish between allocative harms and representational harms. Allocative harms arise when a system withholds certain groups an opportunity or a resource. Allocative harms are the result of a discriminatory allocation—both direct and indirect, and thus the focus of the research efforts that look at fairness as non-discrimination (section 4.2).

Representational harms are from systems reinforcing the subordination of groups regarding race, gender, etc. (Barocas, Hardt, et al., 2017). Representational harms arise when statistical patterns in the training corpora become embedded in semantic representations. These harms are: denigration, stereotyping, recognition, ex-nomination and under-representation. *Denigration* is the use of culturally or historically derogatory terms (e.g. an image tagger that tags black people as 'gorilla')

Stereotyping means the reinforcing of existing societal stereotypes. For example, stereotypical word embeddings analogies such as 'woman is to *nurse* as man is to *surgeon*' or 'woman is to *housewife* as man is to *shopkeeper* (Bolukbasi et al., 2016))

Recognition bias refers to the system's inaccuracy for a certain subgroup resulting in a group being 'erased' or made invisible by a system.

Ex-nomination—or 'un-naming'—refers to "naturalizing existing arrangements of power" (Hartley, 2003, p.86) or in other words something 'goes without saying'. For instance, whiteness has been ex-nominated or made invisible in Western culture, having become the norm Dyer, 1997. An example of ex-nomination there are instances of facial recognition software that do not process darker skin tones (Buolamwini & Gebru, 2018).

Under-representation is the disproportionately low representation of a specific subgroup. e.g. Image search of the term 'CEO' returns mostly white middle-aged men (Kay et al., 2015).

4.3.1. Fair representation practice

Research on the mitigation of representational harms is still in its infancy. Solutions exist mostly in Natural Language Processing (NLP), for representational biases of gender and race. Blodgett et al. (2020) perform a review of 146 bias-related papers in NLP, of which a majority is related to representational harm techniques. T. Sun et al. (2019) performs an extensive review of detection and mitigation methods for gender bias, creating a taxonomy of measures. According to T. Sun et al. (2019), methods to detect representational biases are: Generating test cases, analyzing gender sub-spaces, measuring performance difference across groups. Reducing representational biases in NLP, is done through three types of methods: debiasing the training corpora, debiasing in the word embedding, or adjusting the algorithm. We refer to T. Sun et al. (2019) for details of mitigation methods.

In computer vision (CV), representational harms are underresearched (Z. Wang et al., 2019). Allocative harms that arise due to biases (e.g. by a higher accuracy for majority groups), may lead

to representational biases. Despite the lack of a taxonomy of representational bias detection and mitigation—a computer vision analog to T. Sun et al. (2019)—we can infer that these methods will follow a similar categorization. Thus, in CV we have found two design practices. First, debiased training corpora: FairFace (Kärkkäinen & Joo, 2020, *under review*), a novel face image dataset that is balanced for race, gender and age (similar accuracy across groups). Second, besides a debiased training corpus, an adjusted algorithm: Tang et al. (2020) provide an altered benchmark dataset for gender bias (COCO-GB) and a captioning model to reduce gender stereotypes (e.g. describing a person as a woman because of a kitchen in the background, associating a baseball with men).

4.4. Fairness beyond allocation and representation

The ML-fairness research community is shifting its focus from bias mitigation methods to other aspects that contribute to the fairness of ADM. We briefly discuss those different types of fairness in this section.

4.4.1. Procedural fairness

The decision-making process is found to be an important factor behind fairness judgments Lee, Jain, et al. (2019). However, few design practices exist for ensuring procedural fairness. A possible explanation for this is that organizational policies and processes have to be procedurally fair before being automated (Robert et al., 2020).

Grgić-Hlača et al. (2018) propose the *process fairness* of a classifier $\hat{y}$ to be the fraction of all respondents who consider the use of every feature in X to be fair. In order to crowdsource feature selection, it requires human judgment on a number of user queries. Procedural fairness measures:

1. feature-apriori fairness (judgment without knowing outcomes),
2. feature-accuracy fairness (judgment with knowing a feature improves accuracy)
3. feature-disparity fairness (judgment with knowing a feature increases disparity in outcomes)

Lee, Jain, et al. (2019) adapt procedural justice concepts from organizational justice as design principles to make algorithmic decision-making more fair, but do not provide design practices to do so. However, they do provide an example of interface design in the context of fair division algorithms.

Another procedural fairness issue is that of *contestability*. Contestability refers to the right to contest decisions in the General Data Protection Regulation (GDPR) (European Union, 2016b). Almada (2019) proposes a contestability-by-design approach that integrates design practices for contestability.

4.4.2. Problem formulation

Passi and Barocas (2019) argue that the fairness of ADM depends on the problem formulation. For instance, a loan decisioning system can be framed as a matching problem or as a classification problem, both having different implications for the ML system that solve that problem. This argument is supported by Obermeyer et al. (2019) dissect the problem formulation phase of a prediction algorithm that exhibited racial bias. Martin et al. (2020) propose to use community-based system dynamics—a participatory design method—to achieve a problem formulation and to model long-term dynamics of systems while incorporating the perspectives of stakeholders.

In a similar vein, Selbst et al. (2019) argue that the social context is abstracted away in technical system, and they formulate five 'traps' that designers may encounter and should reflect upon. Designers should determine whether the technical solution:

1. "is appropriate to the situation in the first place, which requires a nuanced understanding of the relevant social context and its politics" (Solutionism),
2. "affects the social context in a predictable way such that the problem that the technology solves remains unchanged after its introduction" (Ripple Effect);
3. "can appropriately handle robust understandings of social requirements such as fairness, including the need for procedurality, contextuality, and contestability" (Formalism);
4. "has appropriately modeled the social and technical requirements of the actual context in which it will be deployed" (Portability); and

5. ”is heterogeneously framed so as to include the data and social actors relevant to the localized question of fairness” (Framing)

4.4.3. Fairness through transparency

Transparency refers to the understandability of a specific model and can provide accountability for potential unfairness (Lepri et al., 2018). There are three different fields of study: Explainability, understandability and interpretability. *Explainability* refers to the right to explanation (European Union, 2016b), protecting the interests of the individual who is subject to the decision. *Understandability* refers to the ”technical skills necessary to understand the underpinnings of algorithms and machine learning models built from data” (Lepri et al., 2018, p. 9). *Interpretability* addresses the opacity that arises in complex ML models such as Deep Learning models. Since the focus of our review is on biases, we refer to Lepri et al. (2018) for a detailed overview of techniques to improve transparency and accountability.

4.4.4. Fairness through human involvement

A number of design practices for the involvement of humans have been proposed. One possible solution might be to include a diverse group of people in the design of the system. Kuhlman et al. (2020) make the case for diversification of workers and practitioners—i.e. representation of important subgroups—to deal with the gender and racial disparities in computing.

Another option is the aggregation of stakeholders’ individual models of fairness. Lee, Kahng, et al. (2019) demonstrate WeBuildAI—a participatory design method used to create a fair allocation algorithm. Srivastava et al. (2019) provide a descriptive ethics approach to discover which notion of fairness humans find most important, emphasizing ”that there is no simple, widely-accepted, normative principle to settle the ethical problem of algorithmic fairness” (p. 4).

Also, fairness might also be achieved if ADM systems defer the decision to a human judge when it is not confident enough to make a decision. De-Arteaga et al. (2020) highlight the risks of full automation, suggesting that a way for humans to keep control over an algorithm is deferment. (Madras et al., 2018; Mozannar & Sontag, 2020) propose methods for algorithms to learn deferring the decision to experts if the prediction is below a certain level of confidence.

4.5. Mapping to the fairness conceptualization

The aspects of fairness that are embedded in existing design practices, are found by relating the design practices (sections 4.2–4.4) to the aspects of the fairness tree (section 3.2). This mapping of *fairness norms* to *fairness design practice* is shown in table 4.3.

4.5.1. Gaps in the mapping

The gaps (marked with a ’-’) in table 4.3 are analyzed as to whether they are potential future research directions.

Consistency across time, in an ADM context means that changes to the dataset possibly redraw decision boundaries. Future research may how decisions by ADM systems may remain consistent over time when retraining on new or updated data. A potential solution may be found in research on adversarial attacks.

Target representation As research on representational harms is relatively scarce (Barocas, Crawford, et al., 2017), we identify the possibility to help ML practitioners specify their target representation of subgroups. Whereas there are methods that claim to remove stereotypes from word embeddings, there is no tractable way to optimize for a certain representation (it should not necessarily be equal representation).

Impartiality refers to ensuring that decisions are made impartially. This raises the question to what extent ADM might be impartial in the sense of Impartiality may require a ’trivial’ solution to avoid tampering with the system, such as audit trails, where all mutations to the system are recorded.

Lawfulness has the ’trivial’ solution of legal domain analysis for the discovery of context-specific laws. Data protection laws—we assume GDPR—will apply to an ADM in a specific setting, but these are already (partially) reflected by the fairness norms.

Deception has received no attention from the research community, as forms of deception (e.g. bluffing, misrepresentation of position, falsification) are human behaviours. As long as AI does not

Table 4.3: Mapping between ADM design practices in this chapter and the fairness conceptualization from chapter 3. Software design practices contain references to the sections in which they are described.

Fairness norms			Fairness design practice category
Fairness dimension	Concept	Specification	
Procedural	Control	Process control	Process fairness (4.4.1), Aggregating individual models (4.4.4)
		Decision control	(Lee, Jain, et al., 2019) (4.4.1)
	Consistency	Across individuals	Individual fairness (4.2)
		Across time	-
	Bias Suppression	Impartiality	-
		Bias suppression	Fairness as non-discrimination (4.2)
	Accuracy	Internal data quality	Data bias (4.1)
		External data quality	Causal reasoning 4.2)
		Model performance	Selection of performance metric (4.2)
	Correctability	Review	Contestability (4.4.1)
		Contest	Contestability (4.4.1)
		Human oversight & correction	Deferment (4.4.4)
	Representativeness	Subgroup involvement	Diversification, participatory design (4.4.4)
		Target representation	-
	Ethicality	Lawfulness	-
		Justification	Problem formulation (4.4.2)
Distributive	Distributive norms		Desired bias (4.1)
	Characteristics		Effort & circumstance (4.2.4), desert, need (4.2.3)
Informational	Justification	Reasonableness	Explainability (4.4.3)
		Timing	Explainability (4.4.3)
		Specificity	Understandability (4.4.3)
	Truthfulness	Transparency	Interpretability (4.4.3)
		Deception	-
		Derogatory judgments	Deferment (4.4.4)
Interpersonal	Disrespect	Inconsiderate actions	Recognition, ex-nomination, under-representation (4.3)
		Abusive actions	Stereotyping, denigration (4.3)
	Privacy	Unwarranted disclosure	-
		Intrusiveness	-
		Use of personal information	Fairness through unawareness (4.2.1), Fairness under unawareness (4.2.2)
	Exposure	Mental Integrity	-
		Physical Integrity	-

show these human behaviours, deception is only an interesting philosophical research direction.

Unwarranted disclosure is related to information security, for the sake of respecting the individuals' privacy, of which we believe this is an already well-explored research direction. A trivial solution is *informed consent*.

Intrusiveness refers to the data collection being as non-intrusive as possible. Although not entirely related to fair machine learning, researchers might investigate solutions for data collection through conversational AI in the field of Human-Computer Interaction.

Exposure to personal dangers Automatic decision-making systems have the ability to violate an individual's physical and mental integrity. Examples of ADM are autonomous weapons—that by definition—expose individuals to personal dangers, or decision-making system subjecting the individual to high levels of stress. Designing AI for mental and physical integrity is important (High-Level Expert Group on Artificial Intelligence, 2019), but there are no design practices for this yet. We refer to Aizenberg and van den Hoven (2020) as they present a research roadmap based on the EU Charter of Fundamental Rights, where they focus on human dignity and fairness. Integrity is implied by the notion of human dignity (breaching one's physical integrity also breaches their human dignity).

4.6. Discussion

The data mining, knowledge engineering, and machine learning communities have produced many ways to make ML systems more fair, assuming fairness to be non-discrimination. However, until now, these fields of study seemed to be disjointed islands.

The mapping is useful in two directions: 1) The mapping functions as a design guide when designing for norms; and 2) It functions as a framework that unifies the different fields of study (e.g. definitions of fairness with respect to stereotypes, problem formulation, non-discrimination).

Furthermore, we have identified literal gaps in the mapping, that are research gaps that offer the fair ML community potential for cross-disciplinary research. These are analyzing and optimizing for consistency across time, optimizing for a target representation, less intrusive data collection, designing AI for mental and physical integrity.

5

Software Engineering for Values

In this chapter, we present the Software Engineering for Values (SEfV) framework, our main contribution (section 5.4), and how it has been established. The chapter further describes in detail the design methodology (section 5.1), the objectives that the SEfV framework should achieve (section 5.2), and the design and development of the SEfV framework (section 5.3).

5.1. Design methodology

Design science (DS) is of importance for creating successful artifacts, and is increasingly applied in information systems research. DS research is the creation and evaluation of IT artifacts intended to solve specific identified organizational or societal problems. Artifacts are not necessarily software, but may be formal logic, mathematics, algorithms, models or even natural language descriptions. In general, they are potentially constructs, models, methods or instantiations.

DS research distinguishes itself from 'regular' design as it requires the researcher not to just solve a problem, but to do this with scientific rigor. For a new artifact to be built with scientific rigor, the researcher should also evaluate the design thoroughly and contributing to theory. Utility implies contribution. The researcher must present evidence to argue what utility is provided, and what demonstrates that utility. DS research relies on existing theories. which can be applied, tested, modified and extended (Hevner et al., 2004).

We develop the SEfV framework according to the Design Science Research Methodology (DSRM), put forward by Peffers et al. (2007), as it gives IT researchers a process for successfully performing DSR. DSRM is a process model that integrates DS principles and practices. The methodology is designed to be commonly accepted, by synthesizing prior literature on DS practices (Peffers et al., 2007), Moreover, it is a proven methodology in information systems research (Peffers et al., 2012). DSRM provides us with a clear procedure for developing the SEfV framework with scientific rigor.

The DSRM process consists of 6 activities:

Activity 1 - Problem identification and motivation The goal of this activity is to define the research problem and to justify the value of the artifact, because DS research should address important and relevant problems (Hevner et al., 2004).

This activity is the starting point of this study. Both the problem and the motivation are addressed in the introduction (section 1.1), followed by a survey on state-of-the-art knowledge about what is possible.

Activity 2 - Definition of the objectives for a solution This activity requires the researcher to infer the objectives for the artifact from the problem definition and knowledge about what is possible and feasible. The objectives for the SEfV framework are derived from the problem statement (section 1.1) and knowledge based on related work (section 2.3). The objective definition is found in section 5.2.

Activity 3 - Design and development During this activity, the researcher determines the desired functionality and architecture through deriving requirements, and then creating the artifact.

The necessary SEfV artifact is a framework, distinctly different from a methodology. A framework is defined as *"a logical structure for classifying and organizing complex information"* (Council, 1999) and is more loose and flexible than a methodology (which is prescriptive) (Wood, 2013). In our case, it not only contains a value specification, but also a design process and guidelines on how to apply the framework on a case. A framework is preferred because it should be applicable to multiple contexts, each with its own context-specific version of the value concepts. The artifact should guide the user to a context-specific solution, and not prescribe what the value is.

Activity 4 - Demonstration An important activity is to show how the artifact is used by solving an instance of the problem. This activity corresponds to the technical investigations aspect of VSD: adapting a value to the context, finding out how it applies to a case and iteratively improve the value specification.

We demonstrate the use of the SEfV framework in chapter 6, for a hypothetical automated loan decisioning system.

Activity 5 - Evaluation During this activity, the observed results of the demonstration are compared to the objectives. Evaluation is crucial in DSRM (Peffers et al., 2012), or as Venable et al. (2012, p. 425) put it: *"evaluation is what puts the "science" in 'design science'"*. The key purpose of evaluating an artifact is to establish its utility and efficacy. The evaluation strategy of this study is described in chapter 7.

Activity 6 - Communication Communicating the artifact to the relevant audiences, emphasizing the problem and motivation, the artifact, and the evaluation results (utility, novelty, rigor, and efficacy) (Peffers et al., 2007).

In the concluding chapter (ch. 8), we emphasize to what extent the artifact contributes to science.

5.2. Objectives

From our problem analysis (section 1.1), research goal (section 1.2), background section on Design for Values (section 2.2), we identify objectives that the framework should achieve in order to be useful for SEfV. The objectives are related to practicality of the framework, engineering design for values criteria, and principles for value-sensitive software engineering. Table 5.1 gives an overview of the framework objectives.

5.2.1. Objectives for the sake of practicality

Practicality refers to the framework being useful in the practitioner's software design process, regardless of their environment or knowledge level. This corresponds to the objectives *agnosticism* and *guidance*.

Agnosticism *"The artifact is software development method agnostic."*

Practitioners use various development models for developing automated decision-making (ADM) systems. In order to be practical regardless of the environment, the artifact must be software development method agnostic: the framework activities must be embedded in any method.

Guidance *"The artifact guides the user through the design activities."* The usability of the framework depends on how well the user is guided through the process. Different demographics of practitioners will be involved in the software design. The software design process may include data scientists, researchers, software engineers, project managers, domain experts, etc., each having different knowledge on the areas that the framework touches.

5.2.2. Objectives for the sake of engineering design for values

As established in the problem statement (section 1.1), the artifact should embed the activities proposed by van de Poel (2015b): *specification*, *translation* and *choice*. We do not include the *discovery* and *verification* criteria, as we consider that future work (see section 1.2).

Specification *"The artifact supports in the specification of general values in terms of design requirements."* In order to achieve its core functionality of translating values to requirements, the framework should be useful in helping the user elicit context-specific value requirements, given a value conceptualization. Thus, this activity assumes a given value conceptualization—more specifically, a value hierarchy as described in 2.2).

Translation *"The artifact supports the translation of such requirements into design parameters."*

This objective relates to technical investigations: clarify how values are persisted into the design.

Here, we have deviated from the translation criterion of van de Poel (2015b) by replacing "engineering characteristics" by "design parameters". Given a high-level architecture model of components and parameters, design parameters are all the 'customizable' aspects of the high-level architecture. For instance, the objective function for an ML model, the data processing procedure.

First, software is an immaterial product, so engineering characteristics—which are the measures that may increase or decrease the quality of the product—correspond to software quality metrics maintainability, performance, safety, interoperability, reliability (IEEE Computer Society, 2014). These are not suitable for translating value requirements into the design. We acknowledge that ADM do not necessarily have to be pure software, for instance, when this framework is applied in robotics there will also be material characteristics like weight, dimensions, degrees of rotation, etc.—but these may be described as design parameters (e.g. an 'actuation' component with these characteristics).

Second, The key idea is that the value is persisted into the design of the system. Thus the goal of the SEfV framework is not to achieve the highest quality—but to make traceable, balanced moral decisions: where in the design the value is (or can be) embedded. This corresponds to the idea of technical investigations in Value Sensitive Design: The artifact embedding the values that are important to stakeholders.

Choice *"The artifact supports the making of design decisions and the clarification and addressing of conflicts between values, in cooperation with the stakeholders."*

Choice relates to two different activities, both of which need to include stakeholders. First, it relates to the activities of identifying and resolving conflicts between values. There are possibly inherent trade-offs between multiple factors within the conceptualization (e.g. accuracy, privacy, fairness, consistency). Moreover, some moral requirements may not be up for compromise ('hard requirements', see section 2.2.2).

Second, given a software requirements specification document, a software design must be chosen to satisfy those requirements.

5.2.3. Objectives for the sake of value-sensitive software engineering

In section 2.3, we described six principles for (software) design methods in order to avoid the negative consequences of value violations (C. A. Detweiler et al., 2010). Most principles focus on the handling of value tensions and including stakeholders. By integrating the principles into the objectives of the SEfV framework, we make sure it avoids the negative consequences of value violations. Since principles 2, 4 and 6 are superseded by the translation and choice objectives, we leave those out and add the remaining principles as objectives: stakeholders, negotiation, and explicitness.

Stakeholders *"The artifact supports the elicitation of values of all stakeholder in as far as relevant for the system under design."*

This objective relates to the identification and engagement with both direct and indirect stakeholders (section 2.2.1).

Negotiation *"The artifact supports the discussion of conflicts between values of the designers and those of the stakeholders, with those who issued the order for the system"*

While capturing the same idea as the previous objective, this objective relates to involving stakeholders in the negotiation of value conflicts, similar to requirement negotiation in SE.

Explicitness *"The artifact supports the justification and evaluation of design decisions in terms of explicit (instantiations of) stakeholders' values."*

Explicitness supports the auditability of value judgments behind design decisions as these are not always clear. In line with DfV (see section 2.2), design requirements should be backtracked to an explicit value judgment. Moreover, as argued in section 2.2.3, the specification of a norm into a requirement is a value judgment in itself. The framework should let the user explicate all these design decisions and trace back values to the design.

Table 5.1: The objectives that the SEfV framework has to achieve

Challenge	Objective	
Practicality	A. Agnosticism	The artifact is software development method agnostic.
	B. Guidance	The artifact guides the user through its design activities
Engineering Design for Values	C. Specification	The artifact allows the user to elicit context-specific operational design requirements from a value conceptualization
	D. Translation	The artifact supports the translation of such requirements into design parameters.
	E. Choice	Support the making of design decisions and the clarification and addressing of conflicts between values, in cooperation with the stakeholders.
Principles for value-sensitive software engineering	F. Stakeholders	The artifact supports the elicitation of values of all stakeholder in as far as relevant for the system under design.
	G. Negotiation	The artifact supports the discussion of conflicts between values of the designers and those of the stakeholders, with those who issued the order for the system
	H. Explicitness	The artifact supports the justification and evaluation of design decisions in terms of explicit (instantiations of) stakeholders' values.

Since most of the objectives relate to functions the framework should provide, we can derive a high-level conceptual model of the framework. See figure 5.1. The model consists of four activities: stakeholder analysis, specification, translation, and choice. In our model, translation and choice go together: By choosing a design practice to satisfy a requirement (e.g. constraining the optimization process of a Machine Learning system to minimize the amount of false positives), the documentation of this solution contains the translation of the value into the design.

We also see that the framework will require three elements: a conceptualization of the value (or multiple values) that we wish to specify, a general architecture with design parameters, and a knowledge base of design practices to aid the design choices. Finally, the framework produces a number of outputs: stakeholders and their needs, design requirements as the product of specifying the value, and a high-level design of the system.

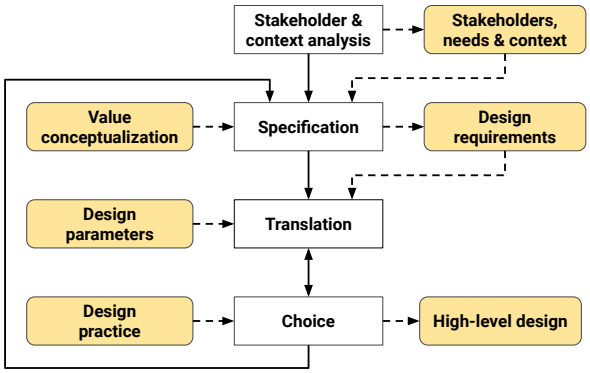


Figure 5.1: A high level view of the SEfV framework as implied by the objectives (solid lines represent activity flows, dashed lines represent information flows).

5.3. Design & development

In this section, the design requirements are formulated (section 5.3.1) and the framework is developed in order to satisfy those requirements (section 5.3.2).

5.3.1. Design requirements

While the objectives describe *what* the framework should achieve, requirements describe as to *how* it should achieve them. Let us describe the translation per objective, for an overview of the requirements, see table 5.2.

Agnosticism In order to be software development method agnostic, we require the framework to be based on the general Requirements Engineering process *(Requirement A1)* and Software Design process *(Requirement A2)*, as described in section 2.1.

These two processes are general Software Engineering (SE) methods, specifying what is expected (requirements) and how the system will satisfy those requirements (software design) (IEEE Computer Society, 2014). Besides SE development methods, ADM are deployed in Data Science, where the Cross-Industry Standard Process for Data Mining (CRISP-DM) is the reference process (Vogelsang & Borg, 2019). The CRISP-DM development model has an initial step named 'business understanding', but "neither the RE nor the ML community [...] detailed what these steps demand" (Vogelsang & Borg, 2019, p.2). As a result, we will assume that data science—despite the analytical process—also benefits of a Requirements Engineering and Software Design framework.

Guidance The framework is responsible for providing sufficient guidance such that other users than software engineers are able to elicit requirements. In order to achieve sufficient guidance, we require the framework to

1) inform about the design stages of the framework *(Requirement B1)*, It guides the user the design processes, as it might not be straightforward what to do.

2) inform about the necessary actions of each design stage *(Requirement B2)*. Not many activities can be automated, so in order for the framework to be as effective as possible, the user should know what is expected of them in each stage.

3) inform about the output of each design stage *(Requirement B3)*. In each stage, the input is modified in a certain way. By informing the user about this modification, they have a better understanding of what they can do with the output.

Specification Given that the framework assumes a value hierarchy (values translated to norms), the framework should provide the abilities to 1) mark which norms do not apply in the given context *(Requirement C1)*; and 2) elicit context-specific software requirements based on each applicable norm *(Requirement C2)*; based on an analysis of the context and which role the system plays within that context *(Requirement C3)*.

The design decision to mark something as non-applicable should happen bottom-up and with a good justification. The main motivation for this requirement is that it reduces the amount of norms in further activities.

The specification of requirements cannot be done automatically, but the user can be guided towards a right specification (van de Poel, 2013).

Translation The framework is required to provide traceability, thus providing the ability to link requirements to design parameters *(Requirement D1)*. This is a direct response to the objective 'translation', as this feature makes it easier to see where requirements influence the design and, more importantly, the value judgments behind the design.

Choice The framework shall provide the ability to identify and to resolve conflicts *(Requirement E2)*.

However, the user must first deal with the issue of value (in)commensurability. Value (in)commensurability, as discussed in section 2.2.2, refers to the ability to directly compare two values on a common scale. In order to be able to make direct trade-offs between values, norms or requirements, the designer should decide whether two norms or requirements are commensurable.

Thus, the framework must provide the ability to classify requirements as 'hard' or 'soft' *(Requirement E1)*. For *hard* requirements, it is not morally permissible to make trade-offs on a common scale with that requirement, whereas a *soft* requirement does allow that (regardless of the extent to which a trade-off is made). The classification as hard or soft is also a value judgment in itself, as it captures the assumptions of the designer that a requirement is (non-)negotiable.

As a result, there are two sets among which potential conflicts exist, namely the set of hard requirements and the set of soft requirements. There are three possible conflicts that require different resolutions:

- Two hard requirements demands respecification or innovation, together with stakeholders *(Requirement E3)*.
- A hard and a soft requirement demands respecification of one of the requirements.
- Two soft requirements allow for more direct conflict resolution methods such as cost-benefit analysis, direct trade-off, maximin, etc. *(Requirement E3)*

In order to support in making design decisions to satisfy requirements, the framework shall provide the ability to connect requirements to design practice *(Requirement E5)*.

Stakeholders In order to support stakeholder values, the user needs to have investigated the stakeholders and their values, thus the framework needs to provide the ability for stakeholder analysis *(Requirement F1)*. The framework must then support the user in eliciting value requirements from stakeholders *(Requirement F2)*.

Finally, for showing how requirements and the software design are supported by stakeholders, requirements need to be traced back to stakeholders supporting it *(Requirement F3)*.

Negotiation In order to achieve the objective of negotiation, the framework shall be usable to discuss value conflicts with those who issued the order for the system *(Requirement G1)*.

Explicitness In order to achieve the objective of explicitness, the framework shall require the user to justify all design decisions *(Requirement H1)*. Furthermore, the framework provides the ability to evaluate all design decisions and justifications in one place *(Requirement H2)*.

5.3.2. Framework development

For each functional requirement, several possible solutions are formulated—based on techniques and approaches from Software Engineering, Design for Values and related work (section 2.3). These possible solutions are structured in a *morphological chart*: a matrix that for each function (horizontal axis) shows the possible solutions to meet that requirement (Dym, 1994).

Through iterative prototyping, a framework is developed that integrates at least one solution for each requirement. Since all requirements are functional—meaning that they require the framework to provide a certain function—we can adhere to the architecture implied in figure 5.1. See appendix A for the morphological chart. The resulting framework is described in section 5.4.

5.4. The Software Engineering for Values framework

Software Engineering for Values (SEfV) assumes a given context-free value hierarchy (see section 2.2.3) that conceptualizes the value into a set of norms—corresponding to the result of the 'discovery' phase of van de Poel (2015b) or 'articulation' phase of Aydemir and Dalpiaz (2018).

As can be seen in the conceptual design, shown in figure 5.2, SEfV looks very similar to software engineering, but adapted to account for values. The SEfV framework consists of three phases that are iteratively followed: *stakeholder & context analysis*; *value requirements engineering*; and *software design*.

Phase 1: Stakeholder & context analysis

During the analysis phase, the designer achieves an understanding of the context, the system and relevant needs. This informs the requirements engineering phase, achieving more successful requirement elicitation. A set of questions is meant to guide this activity.

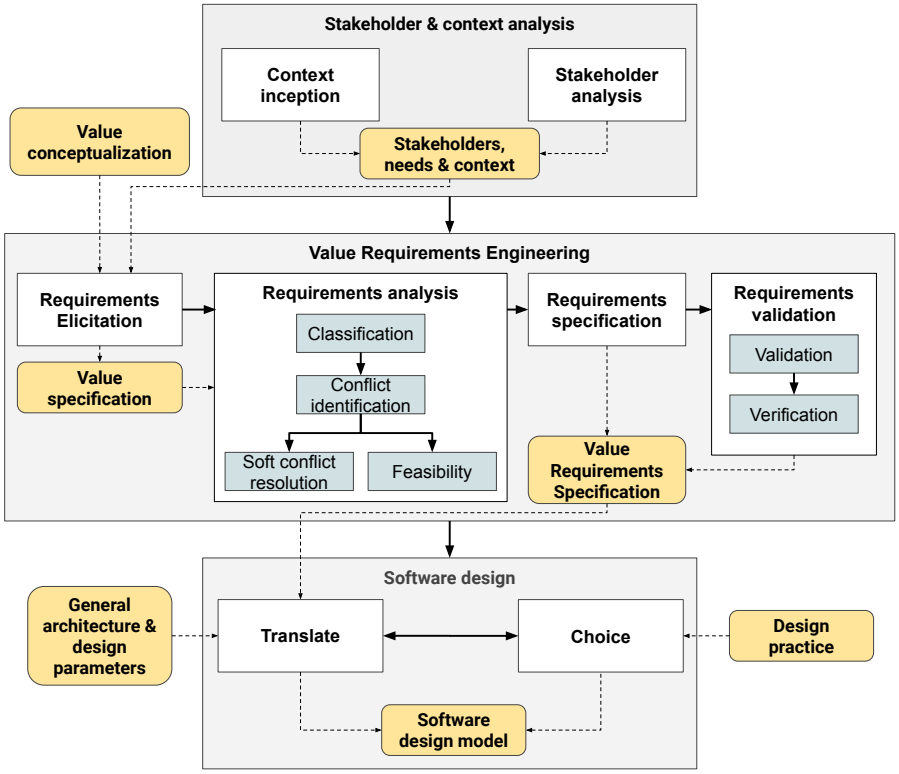


Figure 5.2: Conceptual design of the SEfV framework. The three stages (grey rectangles) are comprised of a number of activities (rectangles) that may take in or produce artifacts (yellow rounded rectangles). Solid arrows indicate activity flows, dashed arrows information flows

Phase 1a: Context inception

- What technical system is being designed?
- What is the social/business process that the system will be embedded in?
- What is the goal of the technical system within the social/business process?

Phase 1b: Stakeholder analysis

- Who interact directly with the system or its output? *(Direct stakeholders)*
- Are there people that do not interact with the system or its output but are impacted or want to have an impact? *(Indirect stakeholders)*
- What are their goals with respect to the system?
- What are their potential benefits of the system?
- What are their potential harms/risks of the system?
- What are concrete scenarios in which the stakeholder will interact or is impacted with the system?

Phase 2: Value Requirements Engineering

During the requirements phase, the designer creates a requirements specification document based on the value conceptualization and the results from phase 1. The phases of value requirements engineering: elicitation, analysis, and validation—resulting in a set of validated context-specific requirements.

Phase 2a: Requirements elicitation

During *elicitation*, the value concepts are made context- and stakeholder-specific. The designer does this by iterating over all the available norms at the lowest level and deciding if and how they apply to the system, potentially through a discussion about the norm. In order to aid the elicitation process, norms should be provided with description, prompt and examples of how they can be implemented.

For each norm, the following questions may help elicit requirements:

- Can the norm be made more specific in:
  1. scope of applicability to the system?
  2. goals or aims strived for *(what the system should achieve to satisfy this norm)*?
  3. actions/means to achieve these aims *(how it is achieved)*?
- In what way does the norm apply to any stakeholder scenario? Can it be described as a user story? ("*As a* `<stakeholder>`*, I want* `<stakeholder need>`*, so that my* `<value>` *is promoted/supported when* `<concrete situation>`").
- Otherwise, why is it non-applicable?

Any elicited requirement is at least provided with an unique id, a name, rationale and is attributed to one or more stakeholders.

Phase 2b: Requirements analysis

During analysis, conflicts between requirements are identified and resolved. It consists of *classification* and *conflict identification & resolution*:

1. Classification as hard or soft requirement

   Hard requirements are incommensurable with other requirements (van de Poel, 2015a) and it is not morally permissible to make a trade-off with this requirement (e.g. complying with laws).

   Soft requirements can be weighed against other requirements. (e.g. The allowed ratio of false positives – false negatives)

2. Conflict identification and resolution

   For each pair of requirements the designer addresses the following:

   - Is there a conflict between these two requirements? Does something prevent from them both being satisfied?
   - If there is a conflict, the designer can do the following:
     - (hard vs. hard) Conflicts between hard requirements need to be investigated for feasibility and at least one of the requirements will have to be respecified, unless a new design option may satisfy both (innovation).
     - (hard vs. soft) How can the soft requirement be respecified such that it does not conflict with the hard requirement anymore? Or does a new design option satisfy both?
     - (soft vs. soft) Depending on the nature of the requirements, the designer can perform a cost-benefit analysis, make a direct trade-off, decide through satisficing or maximizing the lowest scoring value, respecify a requirement or simply prioritize requirements (see section 2.1.1) to take into account for choosing a design. The Analytic Hierarchic Process (AHP) might make it easier to prioritize, since a pairwise comparison is already required for conflict identification. AHP (Saaty, 1990) requires one or more people to specify, on a scale from 0 to 9, how much more important one requirement is over the other? These comparisons can be aggregated into a priority ranking. Moreover, pairwise comparisons are commonly used in the elicitation of value judgments (Lahoti et al., 2019; Srivastava et al., 2019; Yaghini et al., 2019).

Phase 2c: Requirements validation

During validation, the software requirements specification is validated with the stakeholders; and verified for consistency and completeness. The stakeholders should at least be able to answer the following questions positively:

- Do they support the requirements that have been elicited on their behalf?
- Are their goals achieved?

Phase 3: Software design

The design phase assumes the presence of two artifacts: 1) A general architecture of the system consisting of components and design parameters; and 2) A design guide; mapping the norms to possible (domain-specific) practices

During the software design phase, the translation matrix is filled in (table 5.3). The translation matrix maps the requirements (”what?”) to design decisions (”how?”) and design parameters (”where?”).

. For each requirement, the *translation* and *choice* activities are performed simultaneously:

- A solution is deliberated (and justified). Together with the prioritization of requirements, the design guide helps with making a design decision that satisfies the requirement.
- In the relationship matrix, the designer determines to which design parameter the solution is related (i.e. where it is implemented).

5.4.1. Prototype

In order to demonstrate and evaluate the SEfV framework, an instantiation of the SEfV framework has been developed in Microsoft Excel. The user (designing some software system for a value) navigates iteratively through the worksheets. The prototype views are shown in appendix B. See chapter 6 for the application and description of the prototype.

Table 5.2: The requirements that the SEfV framework has to satisfy

Objective		Requirements		
A	Agnosticism	A1	Requirements engineering	Shall be based on the general requirements engineering process.
		A2	Software design	Shall be based on the general software design process.
B	Guidance	B1	Inform stages	Shall provide information about the design processes to perform.
		B2	Inform action	Shall provide information about the necessary actions in each design stage
		B3	Inform output	Shall provide information about the output of each design stage.
C	Specification	C1	Applicability	Shall provide the ability to mark value concepts as non-applicable (together with their child nodes).
		C2	Value specification	Shall provide the ability to elicit software requirements specifying how a norm applies to the system.
		C3	Context analysis	Shall provide the ability to analyze the social or organizational context and the goal of the system within that context.
D	Translation	D1	Traceability	Shall provide the ability to link requirements to design parameters.
E	Choice	E1	Incommensurability	Shall provide the ability to classify requirements that are incommensurable (hard requirements).
		E2	Identify & analyze conflicts	Shall be usable to identify conflicts between requirements.
		E3	Resolve conflicts	Shall be usable to resolve conflicts between 'soft' requirements in cooperation with the stakeholders.
		E4	Feasibility	Shall be usable to resolve conflicts with 'hard' requirements in cooperation with the stakeholders and whether there is a feasible solution.
		E5	Practice	Shall provide the ability to connect requirements to design practice.
F	Stakeholders	F1	Stakeholder analysis	Shall provide the ability to identify direct and indirect stakeholders and their goals.
		F2	Stakeholder requirement elicitation	Shall provide the ability to elicit value requirements from considered stakeholders.
		F3	Stakeholder requirement tracing	Shall trace requirements to one or more stakeholders that support the requirement.
G	Negotiation	G1	Stakeholder involvement	Shall provide the ability to discuss conflicts between values of the designers and stakeholder values with those who issued the order for the system.
H	Explicitness	H1	Justification	Shall provide the ability to justify and explicate design decisions in terms of explicit (instantiations of) stakeholders' values.
		H2	Evaluation	Shall provide the ability to evaluate all design decisions including justifications.

Table 5.3: The translation matrix

		Design parameters	
Requirements	Priorities	Relationship matrix	Design decisions

6

Application: Engineering Automated Decision-Making software for Fairness

In this chapter, the Software Engineering for Values (SEfV) framework is applied to illustrate its potential for translating an ethical value to an auditable design (section 6.2). In order to apply SEfV to fairness to ADM software requirements, a number of system- and value-specific artifacts are necessary:

1. A conceptualization of the value fairness, with descriptions, prompts and examples. For this, the fairness conceptualization from section 3.2 is used, with added descriptions and examples.
2. A guide that maps the fairness conceptualization to possible design practices, meant for aiding the designer in the software design phase. In our case, this is the mapping of the fairness conceptualization to current design practices from section 4.5.
3. A general architecture of Automated Decision-Making (ADM) design parameters – These are established in section 6.1.

Given these three artifacts, ADM systems can be designed specifically for fairness. In section 6.2, this is demonstrated for the illustrative case of a loan decisioning system.

6.1. ADM architecture and design parameters

Our formalization of the ADM architecture is based on the Decision Support System (DSS) architecture by Power (2002). The DSS architecture consists of four components: 1) database component; 2) model component; 3) communication component; 4) user interface component. An ADM differs from a DSS in its ability to decide and enforce the decision. In order to account for this, a 'decision component' has been added. See figure 6.1 for the ADM architecture. The communication component, describing the interconnection between components is not further specified within this study, however it can easily be extended.

6.1.1. Database management component

The database management component describes the configuration of how data is collected, processed, stored and retrieved.

The **data collection** subcomponent consists of the following parameters: *Data sources*, listing which data sources (internal/external) are used, and whether personal data comes directly from the data subject, through observations or has been deduced. *Attributes*, listing all the collected attributes. *Data quality*, specifying (i) the dimensions along which data quality is measured and (ii) an analysis of the collected data along those dimensions (updated after data processing). *Labelling*, a description the (i) target variable, and—if applicable—(ii) the configuration of the data annotation. *Sampling*, describing the (i) sample size and (ii) distribution.

The **data processing** parameter describes how multiple data sources are integrated (*Data consolidation*) and the *cleaning operations* that are performed on the raw data (e.g. imputing missing values, reducing noise, eliminating inconsistencies).

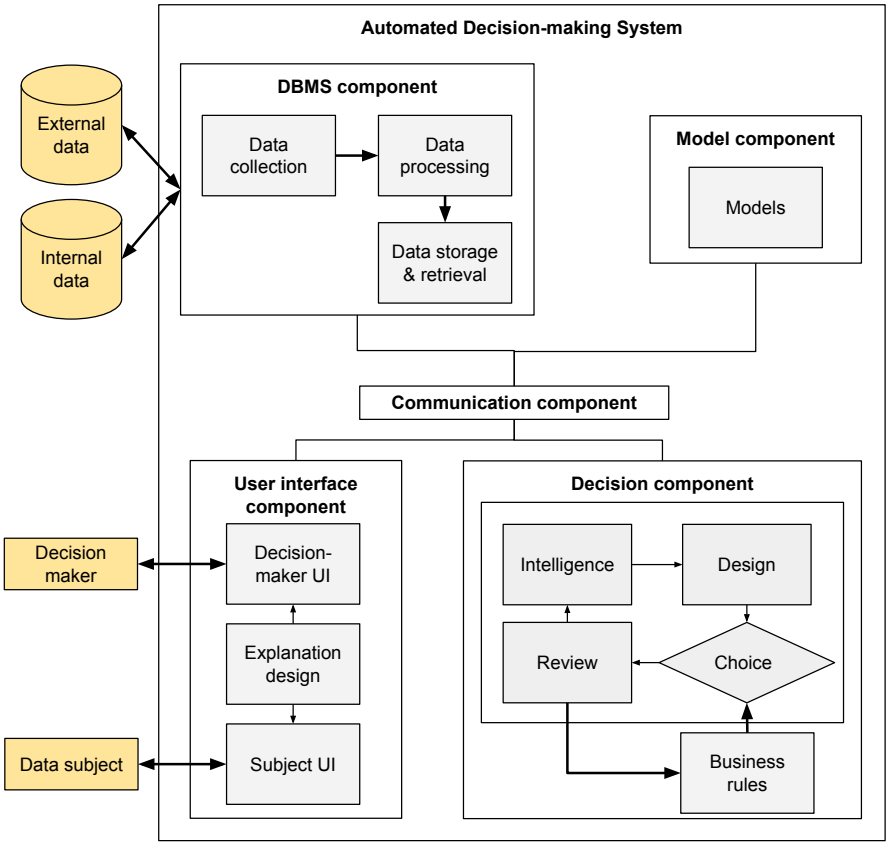


Figure 6.1: A general architecture for data-driven Automatic Decision-making Systems.

The configuration of **data storage and retrieval** is part of the DBMS implementation and is not further specified.

6.1.2. User interface component

The user interface component consists of the user interfaces and explanation design and is the means of communicating decisions to humans. Power (2002) argues that this is the most important component of DSS. For ADM this is not necessarily the case, because not every use case will have a data-subject and subject UI (e.g. ADM in industry). However, ADM will require more from the decision-maker UI for the operator to understand or control the system.

The human decision-maker (or operator/controller) interacts with the ADM through the **decision-maker UI**. It might consist of some form of reporting, details about data subjects, performance measures and override functions.

The data subject interacts with the **subject UI** could show an application form of some kind, decision status, reviewing options, decision and explanation.

The **explanation design** is—within the context of this ADM architecture—the style in which the decision is justified. It informs both the decision-maker and data subject and is based on the decision component.

6.1.3. Decision component

The **decision component** is driver of the decision-making process, coordinating the other components via the communication component. This component is modeled after the human decision-making model by Herbert Simon

Decision-making Decision-making is the process of choosing among alternative options in order to attain a goal (Chiheb et al., 2019). Simon (1977) introduced a—widely recognized (Lewis, 1991)—model of the decision-making process. The process consists of four principal phases:

1. Intelligence: Searching the environment for conditions calling for decision-making (1a). The search process has different characteristics depending on whether it can be structured and whether it is continuous or adhoc.

   During this step, the problem is formulated and the decision-maker forms a mental model of the problem (1b).

2. Design: Understanding the problem (2a), generating (2b) and analyzing (2c) possible courses of action. Solutions are analysed in terms of how they affect their environment.

3. Choice: In a choice procedure (3a), decision criteria are applied in order to choose the best alternative. The choice is communicated to those responsible for implementing it.

4. Review: Assess the outcomes and if necessary, adapt the decision rules, continue the cycle that leads to new decisions.

Simon (1977) also introduced the distinction between programmable and non-programmable decisions. Programmable decisions are repetitive and routine, and a definitive procedure is followed to handling them. Non-programmable decisions are *"novel, unstructured and unusually consequential"* (Simon, 1977, p.46). The extent to which the decision is programmed depends on whether the phases are performed by human decision-makers.

Decision component design parameters Since we adapt the DSS model by Power (2002), we need to move the decision-making from the human to the system. Within DSS, the model by Simon (1977) is commonly referenced. The general decision-making process by Simon (1977) provides a useful high-level basis for the design of the decision-component (see section **??**). Not every ADM will fully automatic perform these phases. In many cases the intelligence phase is performed by the ADM designer, who then programs the design and choice protocols. On the other hand, autonomous vehicles are an example of ADM that autonomously search for a decision to make (e.g. collision detection), design the alternatives and decision criteria, and decide.

In the **intelligence** phase, the problem *search strategy* is defined and the problem is *formulated* (see table 6.1). When an action (application, query, reservation, etc.) triggers the decision-making process, it coordinates the other components towards choosing the best alternative.

Table 6.1: Examples of the intelligence component

Decision type	Problem formulation	Search strategy example
ad hoc binary problem	given individual x with features $[x]$, do we grant x a loan?	triggered upon the receipt of an application, query
ad hoc problem	given individual x with features $[x]$, how much money do we grant x?	triggered upon the receipt of an application, query
continuous	given product y with stock s_y, how many should be bought to meet demand?	daily
continuous	given individual x accessing a website, do we advertise our product to them?	every 10 ms

The **design** phase either explores the solution space with a generation algorithm or it scores pre-programmed alternatives. For the loan example, the generated alternatives might be A=*loan with 20-year payment plan*, B=*loan with 10-year payment plan*, C=*no loan*. Feasibility study might rule out option A depending on company policy, leaving the choice between options B and C.

During the **choice** phase, the ADM applies its (programmed or learned) *business rules* to the evaluated alternatives in order to make a decision. Afterwards, the decision may be communicated to the system/human responsible for effectuating the decision (e.g. revoking a person's drivers licence, denying a loan application). For the loan example, this could mean a simple rule like `if` $creditscore < 690$ `then` $noloan$, or `if` $confidence$ `then` $defer$, deferring the decision to a human.

During the *review* phase, the outcome is assessed and necessary *changes* are made. For example, decision rules are changed, models are retrained.

The decision component highly interacts with the other components. It communicates with the outside world (decision-maker, data subject, auditor, etc.) via the UI component. In the intelligence phase, data such as product stock and price might be necessary to formulate the problem. In the choice phase, it pulls data from the database component and requests predictions from the model component in order to choose an alternative (e.g. the probability of defaulting given features $[x]$) and the decision is communicated to other systems and to the manager (and data subject) via the UI. In the review phase it may decide to communicate with the model component to update (retrain) the models.

6.1.4. Model component

The **model component** consist of a set of algorithms and/or statistical inference models. The models use data from the database component to make a prediction for a new data entry (i.e. of the data subject). The component may consist of multiple models, as the decision component may use their predictions to decide. Within the scope of this study, the model component will be a single binary classifier.

In the supervised Machine Learning (ML) setting, with protected attributes $G \in \mathcal{G}$, all additional features $X \in \mathcal{X}$, and true labels $Y \in \mathcal{Y}$. The goal is to learn a classifier $\hat{y} : \chi \times \mathcal{G} \rightarrow \mathcal{Y}$, given a set of training samples $T = \{(x_1, g_1, y_1), \ldots, (x_n, g_n, y_n)\}$. Probabilistic classifiers produce a score $\hat{y}_S \in [0, 1]$ for which a scoring function $S : \hat{y}_S \rightarrow \mathcal{Y}$ maps the score to a classification for some threshold $t = [0, 1]$. The focus of this work is binary classification, so $\mathcal{Y} = \{0, 1\}$, where $y = 1$ (e.g. receiving a loan) corresponds to a *favorable* outcome and $y = 0$ to an *unfavorable* one.

During the ML development, more parameters of the ML model design are specified. The ML development pipeline generally consists of the following phases (Koen, 2019): 1) understanding, 2) preprocessing, 3) model generation, 4) model training, 5) postprocessing, 6) deployment and 7) performance analysis.

Preprocessing Preprocessing generally consists of two steps: *feature engineering* and a *class imbalance correction*. In feature engineering, the raw data (D) is mapped to the features that are used by the algorithm. $f : D \rightarrow X$. This function may simply select a number of attributes and produce a feature vector x, change the encoding of data (e.g. one-hot encoding of categorical data) Feature extraction methods create new features from the data. For instance, interaction terms (e.g. `race` × `gender`) would produce a more fine-grained feature (Kearns et al., 2018). dimensionality reduction (e.g. PCA), automatic feature construction (e.g. word2vec).

Function f describes all these transformations to the raw data. The order in which transformations are applied influences the quality of the features.

In situations where there is a small minority class (e.g. spam filtering, fraud detection, rare diseases), one implements a method to deal with the class imbalance. Ways to correct for class imbalance are penalizing minority misclassifications, optimize for area under curve (AUC) instead of accuracy, SMOTE, anomaly detection.

Model generation The instantiation of the ML model requires the selection of the algorithm, setting the (hyper)parameters, specifying an objective function (including loss function and distance metric) with additional constraints and regularization. Many different classifiers are useful for binary classification. They can be based on Bayes (Naive-Bayes, Bayesian nets), linear (linear discriminant, perceptron, logistic regression, support vector machine [SVM]), nonlinear (multi-layer perceptron, generalized, decision trees).

Many algorithms require hyperparameters to be set (e.g. kernel and slack of SVM, the k in $k-$nearest neighbors algorithm, tree depth). Model parameters define the individual model and can be learned (e.g. regression coefficients, perceptron weights, decision tree split points). The objective function is the function that the classifier optimizes during training. For supervised learning, the goal is to find an optimal mapping function $f(x)$ (the cost function) that minimizes the loss functions of the individual training samples,

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(y^i, f(x^i, \theta)) + \lambda R(f)$$

where N is the number of training samples, θ is the parameter of the mapping function and L is the loss function. (e.g. MSE, SSE, SVM cost function), with additional parameters, constraints and a regularization term $\lambda R(f)$. The loss function measures the error between the predictor $\hat{y}$ and label y. For example, square loss (linear regression), hinge loss (SVM), 0-1 loss, information gain, cross entropy.

Classifiers that minimize distance (e.g. k-NN) use a distance metric (c.q. similarity metric) such as Manhattan, Euclidian, Non-euclidian, Mahalanobis, Cosine and Hamming distance. The regularization term consists of a regularization function $R(f)$ and the compromise parameter λ, penalizing the model complexity to avoid overfitting.

In many applications, algorithms are applied with the *default* setting, where these parameters are not explicitly defined and well-chosen.

Model training The training phase highly affects the model performance. The gold standard, data splitting, optimization method and evaluation metric. The data that contains the ground truth How the data is split (by case, by subject) into training-validation-test sets, and how often ($n-$fold cross-validation using partitions or random draws). In some uses the sampling order is specified (as in curriculum learning). Batch training online iteratively trains on The measures against which the predictive power of the candidate models are evaluated and selected. Besides accuracy, measures are recall, AUC, F-score, etc.

The optimization method specifies how the objective function $f(x)$ is maximized/minimized over training samples X and a stopping criterion. It may be subject to additional constraints. First-order optimization (e.g. gradient descent, AdaGrad, SAG), high-order optimization (e.g. Newton's Method, Natural Gradient), derivative-free optimization methods (e.g. heuristics, coordinate descent).

According to S. Sun et al. (2019), stochastic gradient descent is widely used, but many users adopt them as black box optimizers without setting parameters. They provide an extensive comparison of available methods in order to give an understanding of the properties, parameters and (dis)advantages. For algorithms that don't converge or that overfit, we can check every iteration of the training process whether a stopping condition is satisfied.

Post-processing In post-processing, a trained classifier $\hat{y}$ is adjusted, which produces a *derived classifier* $\hat{Y} = f(\hat{y}, A)$. Examples are: pruning, removing correlations with the sensitive attribute A, deferring.

Deployment and performance analysis The deployed model may consist of multiple trained models, where business rules explicate how the models are used in the decision-making. Design parameters include *Monitoring*, when and how *re-training or on-line* learning takes place. Another free parameter is the way in which multiple classifiers are combined (majority voting, geometric average, boosting).

Finally, we summarize all design parameters in a tree structure. Figure 6.2 gives an overview of the ADM design parameters. The design parameters of the model component is further specified in figure 6.3. The formalization of both the ADM and ML model (section 6.1.4) have intentionally been made distinct, despite an overlap in practice. The aspects of the ML pipeline that have explicitly been attributed to the database component are *data gathering* and *data cleaning*, and *maintenance* to the decision component (in the 'review' phase). It is important to make a distinction between the automated decision-making process at an abstract level, and the technology at the center of the ADM process, such as a supervised machine learning (ML) model. Not only are both the domain of different experts, but both have different characteristics.

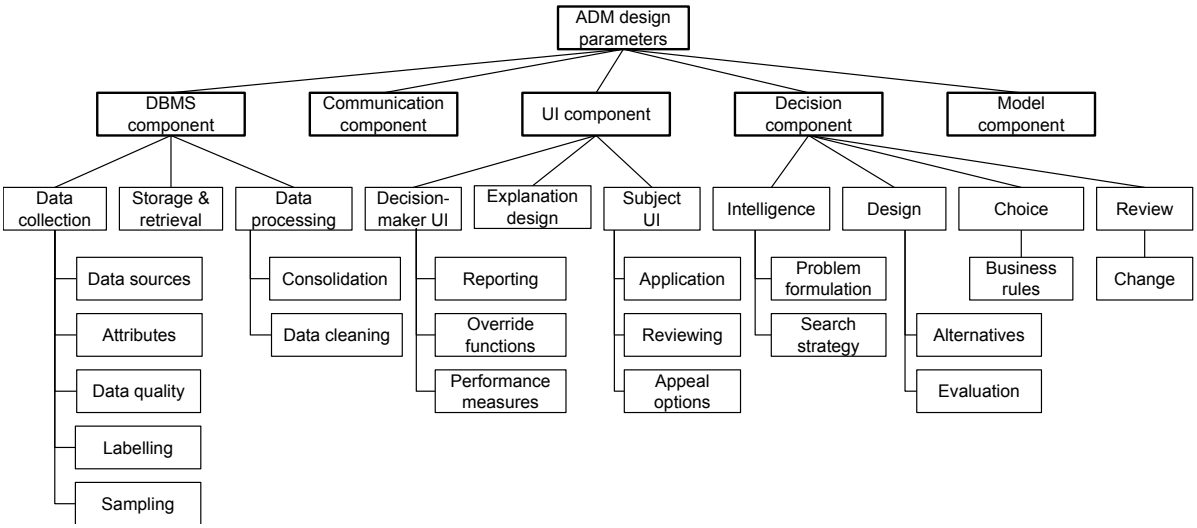


Figure 6.2: Design parameters of a general ADM architecture (the model component is specified in figure 6.3

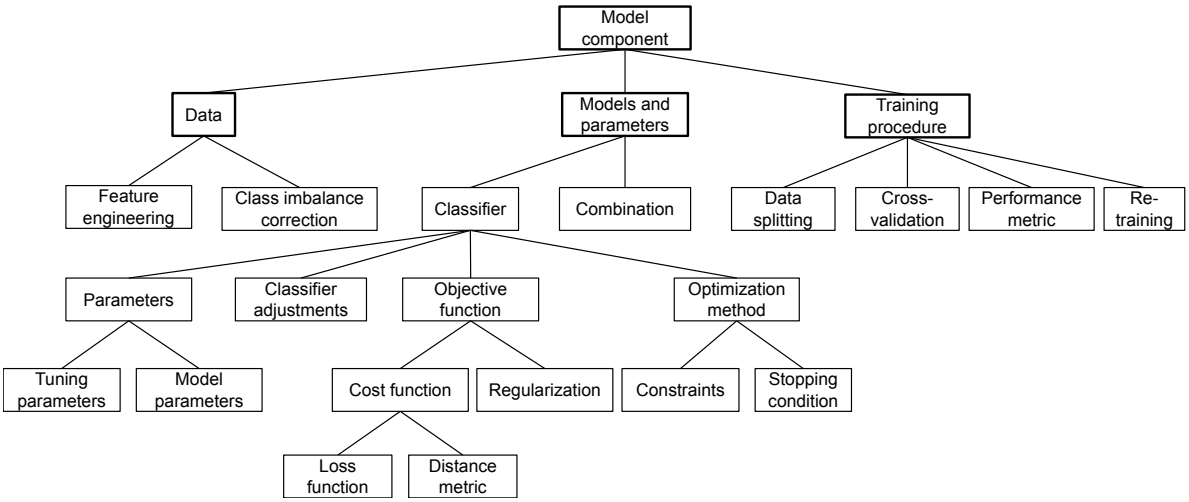


Figure 6.3: Design parameters of the model component in figure 6.2 (a Machine Learning model for the binary classification problem)

6.2. Application of SEfV to specify fairness for ADM

Given the SEfV framework and the system- and value-specific artifacts described in the beginning of this chapter, we can demonstrate the use of SEfV. The application of the SEfV framework is demonstrated for the illustrative case of a loan decisioning system, described in section 6.2.1.

6.2.1. Illustrative scenario: Automatic loan decisioning system

The Head of the loans department of a regional bank of ca. 300 employees (e.g. bank tellers, loan officers, bank managers, internal auditors, marketing) wants a system that automatically decides whether or not an applicant is granted a loan.

They receive over 100 loan applications a day, which puts a lot of pressure on the small team of loan officers. If this can be automated, the loan officers can better handle applications that require more attention. Loan officers have multiple tasks: marketing loan products to customers, processing documents for loans and manage debt collection.

Current way of managing loans In order to handle applications, they already use an automatic online loan management system. The head of the department wants it extended so it automatically decide whether an online applicant should get a loan. Currently, they minimize the risk of giving loans

to people that will not be able to pay back the loan. They do this with a manual checklist and a set of rules (e.g. "if $creditriskscore < 690$, then $noloan$").

The automatic loan decisioning system The IT department has made a proof of concept of the loan decisioning system. It calculates the probability that an applicant will default (go bankrupt) on their loan. If the score is below a certain threshold, they are considered a good borrower and the loan is granted.

The system is trained on a dataset[1] of 1000 historical records of the bank, containing the following attributes:

- Loan application information (credit amount, duration, purpose)
- Account information (checking, savings, bonds, existing credits)
- Credit history (earlier requests, instalment plans, instalment rates, other debtors)
- Personal information (age, gender, marital status, employment status, residence, foreign worker status, dependents)
- The recorded outcome (whether they were a good borrower or bad borrower)

The head of the loans department remarked that it doesn't feel fair, but doesn't know exactly why. Moreover, the department head is anxious since recently another bank has been outed as 'sexist' because they had a similar system that apparently gave women lower loans than men[2].

6.2.2. Loan decision context and stakeholders

First, we perform the analyses of phase 1 5.4.

In the context analysis, the loan decisioning system is situated within a social and organizational context. See figure 6.4 for the performed context analysis.

Technical system being designed

An automatic loan application decisioning system, taking customer data as input and outputting whether the individual should receive a loan.

Within which (social/organizational/business) context is the technical system built?

The bank has a loan application procedure, but gets over 100 applications a day. Automating the process gives the staff more time to handle the more complex cases. The procedure must be fair, since reputational/legal risks may arise if the system is sexist or otherwise biased.

Goal of the system

Automatically process loan applications, calculating a credit risk score (good/bad borrower), based on a given set of attributes.

Figure 6.4: Context analysis for the loan decisioning case, performed in the prototype

The stakeholder analysis has resulted in numerous findings: First, besides the direct system stakeholders, also indirect stakeholders have been identified (bank manager, internal auditor). Second, There is a potential benefit for streamlining the auditing process, as the system might integrate an audit view of some sorts. Third, fairness considerations might not only apply to the applicant, but also loan officers, whose tasks will be automated. As a result, some loan officers might be transferred or fired. Fourth, the concrete scenarios that have been derived are: monitoring, auditing, application, approval, rejection, appeal, decision interpretation, and the handling of an appeal. These scenarios are useful for eliciting requirements. The result of the stakeholder analysis is shown in figure 6.5.

[1] German Credit Dataset, from: https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)
[2] based on https://www.wired.com/story/the-apple-card-didnt-see-genderand-thats-the-problem/

#	Stakeholders	(In)Direct	Goal	Potential benefits	Potential harms	Concrete scenarios
1	**Bank manager**	Indirect	Minimize risk of lenders defaulting on their loans.	More efficient loans process, lower costs	Bad publicity for the bank, less clients.	-
2	**Head of the loans dept.**	Direct	Maximize department efficiency	Less work for assessing risk.	Losses due to false positives by the system	Monitoring the system
3	**IT team leader**	Direct	Design and develop a robust, fair and reliable decision system.	Minimise the workload of colleagues. Recognition for sucessful completion of the project.	Design a system with an unwanted behaviour (e.g. similar to the "sexist" behaviour of their competitors)	Maintain the system, monitor the system
4	**Loan officer**	Direct	Minimize time required to process loan application. Satisfied customers.	Stress relief, more quality time with clients. Able to evaluate more loan applications in a set amount of time.	Unable to control the system, Take care of customer of complaints, Some loan officers might become superfluous and this might be unfair for them	Handling an appeal, interpreting the decision, monitoring the system
5	**Applicant**	Direct	Obtain a loan (or maximize their loan opportunity)	1) Time saving by receiving an answer within a matter of minutes. 2) An automated (explainable) decision may increase their trust in the decision	a loan officer (interact with a machine not a human). Failure of the system to provide explanations might lead to confusion.	Apply for a loan, Loan granted, Loan denied, appeal
6	**Internal auditor**	Indirect	Easier auditing process, minimize risk of unfair results and resulting lawsuits or reputational risk.	The system may be designed to support auditability, computer system provides quantitative over qualitatitve information	Lawsuits/reputational loss of bank	Audit the system

Figure 6.5: Stakeholder analysis for the loan decisioning case, performed in the prototype

6.2.3. Value requirements engineering for the loan decision system

First, the designer iterates over the norms and argues if and how it applies to the loan decisioning system. For example, for the norm *decision control*, a possible prompt to guide the specification is shown in figure 6.6. Decision control is considered *not-applicable* as it is argued that the applicant should not decide on the allocation of funds by the bank, both for their own interest (they might not be able to pay back their mortgage if it is too high for them), and bank's interest (since the money is managed by the bank).

Another norm, *process control*, may be translated to two possible context-specific requirements:

1. **Motivation**: "As an applicant, I want to be able to motivate my loan application."

   Rationale: There may be important aspects to my application that are not part of the form.

2. **Cancellation**: "As an applicant, I want to be able to cancel the application before the decision is made, without a negative result for me."

   Rationale: The applicant should be able to try again when they are more certain, without us storing their information.

Second, the requirements analysis phase is performed: classification as hard or soft requirement, conflict identification, conflict resolution.

Classification Requirement 1 (motivation) is classified as a *soft* requirement—we can make trade-offs with it, while requirement 2 is considered a *hard* requirement, as it is particularly unfair to force a person to receive a loan rejection as it has consequences for an individual (it is registered in the applicant's credit ratings).

The *process control* requirements do not conflict with any other requirement. However, there are a number of other conflicting requirements (appendix B)

Resolving conflicts Two hard requirements that conflict are *consent* and *individual consistency*: Consent requires that applicants are able to consent to the use of their information. Individual consistency requires that individuals with similar financial status should receive similar treatment. However, if there are individuals that do not consent the use of some of their personal (financial) information, it is harder to ensure and give evidence of individual consistency.

Since we deal with two hard requirements, we can only respecify or innovate to resolve the conflict. In this case, a solution is the design might not ensure total individual consistency, but to a certain degree. In essence, we treat *individual consistency* as a soft requirement. Another potential resolution

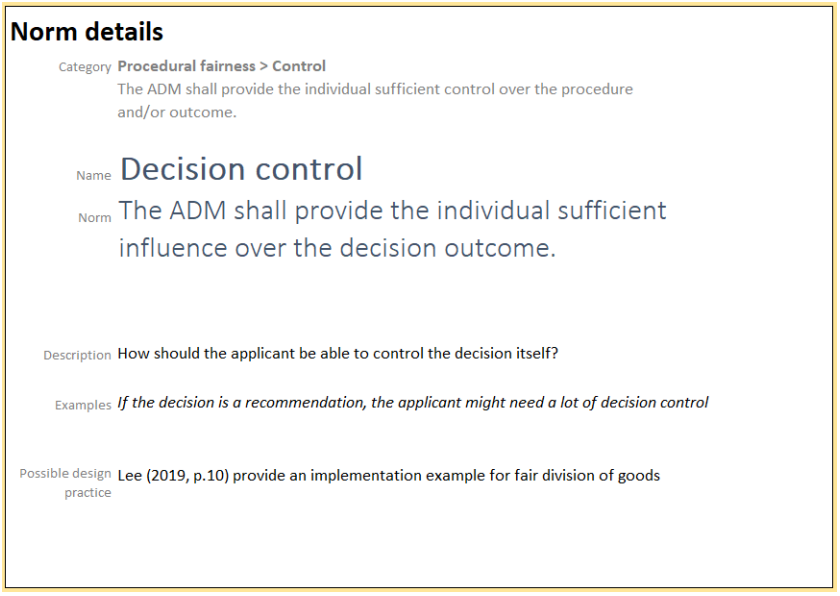


Figure 6.6: Details for the norm *decision control*, as shown in the prototype

is to respecify the *consent* requirement to '...to consent to the use of their non-financial information', as we regard the individual's financial data crucial for the decision.

Another possible conflict arises between the hard *consent* requirement and the soft requirement *credit risk model accuracy*: "As a decision-maker, I want the accuracy of the credit risk scoring model to be as high as possible." They conflict in the sense that withholding data that is sensitive might lower the performance of the model. This can be resolved in a similar way as the previous example. We could, for example, limit the consent requirement to only allow consent for features that score below a certain feature importance. Whether this solution erodes the underlying value too much, is up for discussion with the relevant stakeholders.

There are also two soft requirements where we have identified a conflict. Credit risk model accuracy: Unwanted bias suppression: The system should not use information unrelated to the loan decision for its classification, either directly or indirectly (in the form of a proxy).

Here we can make a direct trade-off, e.g. through weighting both terms in the optimization of the ML model (This weighting can be conveyed through requirement prioritization. Using the Analytic Hierarchic Process (AHP), we can make a pairwise comparison of the two requirements. The aggregation of all these comparisons gives a ranking of requirements, conveying our direct trade-offs as coonflict resolution. Let us decide that *credit risk model accuracy* is fairly more important than *unwanted bias suppression*, since an accurate model will also benefit the applicants, thus on a scale from 0 to 9, we rate it 6.

At this point, in a real-life context, the requirements should be validated with stakeholders. We assume the requirements to be validated and perform the next phase of software design.

6.2.4. Loan decisioning system design for fairness

Some requirements demand hard choices (e.g. for bias suppression we must choose one out of many different statistical notions of fairness), but also trivial solution. We demonstrate this phase of the framework for a trivial and a hard translation.

The software design for the *Motivation* requirement is trivial: The loan application form shall contain a textbox where the applicant can motivate their application. The justification is that at their point of access, the UI should allow the applicants to motivate their application. The corresponding design parameter (the component where it is implemented) is the subject UI.

The software design for *Unwanted bias suppression* is chosen using the design guide B. For the norm *bias suppression*, the guide refers to the Aequitas fairness tree for selecting a statistical measure (described in section 4.2.6). While traversing the Aequitas fairness tree, we make value judgments: We decide that fairness should be based on classification parity. We do not trust the labels to be the ground

truth, since there is no data for people who *would have been a good borrower, had they received a loan*. Therefore, we are advised to use Counterfactual fairness (would an applicant be treated differently if they were of another protected group?) So our solution is to implement Counterfactual fairness (Kusner et al., 2017), with as justification that we want classification parity over statistical parity, but that the labels are not ground truth.

The output of this phase is a list of justified solutions that are traced back to value judgments and fairness requirements. This list is the basis for the implementation of the software. For a formal evaluation of the framework, see chapter 7

7

Evaluation

In this section, we evaluate the Software Engineering for Values (SEfV) framework in its ability to help design ADM software for fairness.

Evaluation refers to comparing the effects of the use of the artifact with selected criteria to conclude whether it is satisfactory (Verschuren & Hartog, 2005). Not only is evaluation crucial in design science (DS), but selecting the right evaluation method emphasizes the utility of an artifact (Hevner et al., 2004). In order to select the right evaluation method, we use two frameworks which provide guidance in DS research evaluation. Venable et al. (2012) guides in choosing an evaluation strategy that matches the context of our study, while Prat et al. (2015) helps in selecting an evaluation method that matches the selected strategy. In section 7.1, the evaluation strategy and method behind the experiments are explained. The experimental setup is described in section 7.2. Results from conducting the experiments are described in section 7.3.

7.1. Evaluation strategy

Usefulness, effectiveness, utility, efficacy are the most prevalent criteria in DS (Prat et al., 2015). *Usefulness* is ”the degree to which the artifact positively impacts the task performance of individuals”, *effectiveness* is ”the degree to which the artifact achieves its goal in a real situation”, *utility* measures ”the value of achieving the artifact’s goal”, and *efficacy* is ”the degree to which the artifact achieves its goal considered narrowly, without addressing situational concerns” (Prat et al., 2015).

We select *usefulness* as key criterion, because the goal of the framework is to help practitioners in performing a task. Furthermore, usefulness over efficacy is in line with VSD as it aims for ”progress, not perfection” (Friedman & Hendry, 2019).

Given the focus on usefulness as opposed to efficacy, an *artificial ex-post* evaluation strategy suits our needs best. Artificial refers to the SEfV framework being tested in a controlled environment, ex-post means that we should evaluate an instantiation of the framework (the SEfV prototype). Although Venable et al. (2012) suggests naturalistic ex-post evaluation (*naturalistic* = in a real-life setting), it has been found infeasible to gain organizational access within the scope of this study. Thus, opting for artificial ex-post evaluation, Venable et al. (2012) guides us towards evaluation through a lab experiment.

We evaluate SEfV through a lab experiment where students and researchers are guided through the application of the SEfV framework to translate fairness into specifications for an illustrative case of an ADM. Participants are afterwards surveyed about how well the framework has achieved its objectives. Given our focus on usefulness as criterion, we opt for the pattern of ’laboratory, student-based evaluation of usefulness’ as opposed to the similar pattern ’practice-based evaluation of usefulness’ (Prat et al., 2015). In the case of SEfV evaluation, we make a quantitative measurement of the participant’s perception of usefulness, where the participants are students and researchers.

7.2. Experimental setup

In this section, the experimental setup is described. What the goals of the experiment are 7.2.1, how those are made measurable 7.2.2, and the followed protocol 7.2.3.

7.2.1. Experiment research questions

Besides usefulness, we evaluate the framework for the following criteria, based on the SEfV objectives (section 5.2): conflict management (objective F), explicitness (obj. H), traceability (obj. D). Furthermore, we evaluate *perceived fairness* in order to assess participants' trust in the software design that they are confronted with. *Perceived fairness* should be evaluated as a measurement of the effect on the real world (i.e. do people perceive the ADM to be more fair?). This follows from the research goal to help practitioners build ADM for fairness.

The following research questions will be answered through the experiment:

1. Do the participants perceive the framework to be useful for performing the tasks? (objectives B-H)

   The following questions require absolute evaluation, since we in our case we cannot measure improvement relative to something else.

2. Does the framework provide satisfactory support in conflict management (obj. E, G)?
3. Does the framework provide satisfactory support in explication of design decisions and justifications (obj. H)?
4. Does the framework provide satisfactory support traceability of requirements (obj. D)?
5. Does the framework provide satisfactory perceived fairness?

Participant selection includes academic staff, students and graduates, preferably familiar with Computer Science. In order to answer the experiment research questions, participants are selected who have at least some familiarity with the skills required for the tasks at hand. Although familiarity is not required, it is assumed to increase the extent to which the task is completed successfully, such that the participant may evaluate it more adequately.

7.2.2. Measurement

In the measurement of evaluation criteria, most items are measured as agreement with a corresponding statement. This agreement is measured through a standard 7-point likert scale for agreement: *Strongly disagree, Disagree, More or less disagree, Undecided, More or less agree, Agree, Strongly agree.*

A number of items relate to the skills of the participant in 1) Requirements Engineering, 2) Software Design, 3) Machine Learning, 4) Fairness in Machine Learning, and 5) Design for Values. This gives an understanding of demographics among the participants. As we do not need a fine granularity in the distribution of responses, we measured skill on a 5-point likert scale of: *No skill, Low skill, Moderate skill, High skill, Very high skill.*

The evaluation criteria are operationalized as follows:

- Perceived usefulness is made up of general statements on the usefulness of every activity that the participant performs: context analysis, reflection, specification, translation, choice, conflict detection, and auditing a design. Reflection refers to reflection on the prompted norm in order to translate it to requirements.
- Conflict management is measured through *Perceived usefulness in conflict detection*, which is already operationalized, and the amount of conflicts identified during the design task.

  Conflict resolution could not be tested in the artificial experiment and therefore is not measured.
- Explication is measured through perceived explicitness as designer and auditor, the perceived justification as designer and auditor, and the number of explications and number of justifications.
- Traceability is measured through the perception of traceability of the requirements as auditor.
- Perceived fairness is measured through a statement after auditing.

Table 7.1: Operationalization of the evaluation criteria and the participant skills. The amount of points on the likert scales written between parentheses

Criteria	Code	Conceptualization	Measurement	Scale	Item / statement
Self-reported skills	SDV	Self-reported skills in design for values	Likert scale (5)	Ordinal	How skilled are you in Design for Values?
	SRE	Self-reported skills in requirements engineering	Likert scale (5)	Ordinal	How skilled are you in Requirements Engineering?
	SSD	Self-reported skills in software design	Likert scale (5)	Ordinal	How skilled are you in Software Design?
	SML	Self-reported skills in machine learning	Likert scale (5)	Ordinal	How skilled are you in Machine Learning?
	SFML	Self-reported skills in Fair-ML	Likert scale (5)	Ordinal	How skilled are you in Fairness in Machine Learning?
Perceived usefulness	PUD	Perceived usefulness (designer)	Likert scale (7)	Ordinal	The tool had a positive impact on my task performance as a designer.
	PUA	Perceived usefulness (auditor)	Likert scale (7)	Ordinal	The tool had a positive impact on my task performance as an auditor.
	PUCA	Perceived usefulness context analysis	Likert scale (7)	Ordinal	The tool helped me understand the context.
	PUR	Perceived usefulness reflection	Likert scale (7)	Ordinal	The tool helped me think about how fairness might apply to the system.
	PUS	Perceived usefulness specification	Likert scale (7)	Ordinal	The tool helped me specify requirements.
	PUT	Perceived usefulness translation	Likert scale (7)	Ordinal	The tool helped me translate requirements into the software design.
	PUC	Perceived usefulness choice	Likert scale (7)	Ordinal	The tool helped me choose a software design to satisfy the requirements.
	PUE	Perceived usefulness explicitation	Likert scale (7)	Ordinal	This tool had a positive impact on my task of explicating design decisions.
Conflict management	CD	Conflict detection		Ratio	of requirement conflicts that the subject has identified during the experiment
	PUCD	Perceived usefulness conflict detection	Likert scale (7)	Ordinal	The tool helped me identify conflicts between requirements.
Explicitness	PED	Perceived explicitness (designer)	Likert scale (7)	Ordinal	I made choices that I otherwise would have made implicitly.
	PJD	Perceived justification (designer)	Likert scale (7)	Ordinal	I motivated choices that I otherwise would not have motivated.
	ED	Explicitness (designer)		Ratio	of design decisions made
	JD	Justification (designer)		Ratio	of justifications given
	PEA	Perceived explicitness (auditor)	Likert scale (7)	Ordinal	The designer made explicit assumptions about what they consider fair.
	PJA	Perceived justification (auditor)	Likert scale (7)	Ordinal	I understand why the designers made these assumptions.
Traceability	PTA	Perceived traceability (auditor)	Likert scale (7)	Ordinal	The framework helped me find the value assumptions behind the software design
Perceived fairness	PFA	Perceived fairness (auditor)	Likert scale (7)	Ordinal	I would consider an automatic decision by this system fair, even if I get a negative outcome.

7.2.3. Protocol

Experiments were conducted according the following protocol:

1. Introduction: The participant is told they will work on a software design for fairness, and audit another design for fairness.
2. Pre-design survey: The participant fills in a survey about their self-reported skills
3. Briefing: The participant reads the case description (from section 6.2.1) and is briefed on the design task and the SEfV prototype. They are specifically instructed to discuss and speak out their thoughts.
4. Design task: The participant is instructed to design a software system for the case, using the SEfV prototype.
5. Post-design-survey: The participant reports their perceptions on designing with the SEfV prototype
6. Audit task: The participant is given a filled in copy of prototype and is asked to audit the design decisions.
7. Post-audit-survey: The participant reports their perceptions on auditing with the SEfV prototype

7.3. Results

Experiments have been conducted among academical staff, students, and graduates of TU Delft (n = 12). Sessions took approximately one hour, following the protocol as described in section 7.2.3.

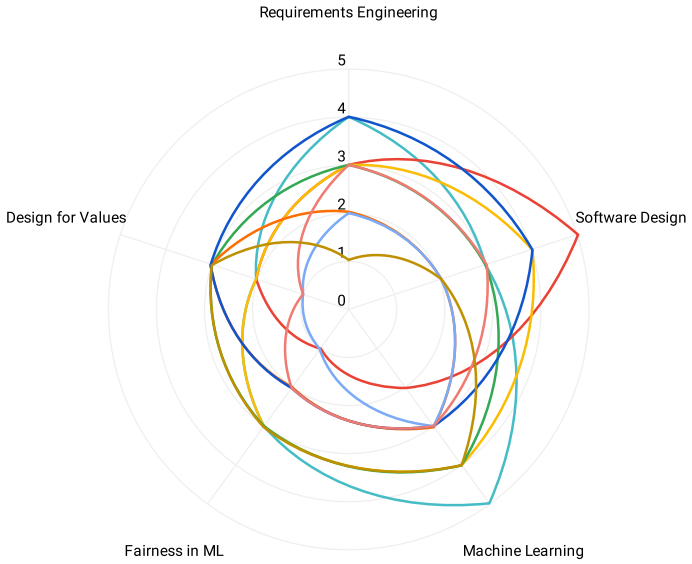


Figure 7.1: Radar diagram of the self-reported skills of each participant (1 = no skill, 2 = low skill, 3 = moderate skill, 4 = high skill, 5 = very high skill)

The artifact has been evaluated from most perspectives, but two viewpoints are underrepresented: those skilled at debiasing or at designing for values. Despite the low number of participants, software engineering (SE) and machine learning (ML) skills are well represented. For requirements engineering a quarter of the participants self-reports as highly skilled, and almost half of the participants is (very) highly skilled in software design (42%) and ML (50%). We also see that the two SE skills go well together, as they are significantly correlated (*Spearman's rho* $= 0.579, p = 0.048$). Self-reported skills in the implicated fields (Software engineering, machine learning, fairness in machine learning, design for values) are shown in figure 7.1. See figure C.1 for more detailed graphs of the response distributions.

Participants reported a high learning curve, but were productive nonetheless. Within the timeframe of around 30 minutes, on average between three and four requirements were specified ($M = 3.8, SD = 0.5$). From these requirements arise on average 17 explicit design decisions ($M = 17.3, SD = 2.6$).

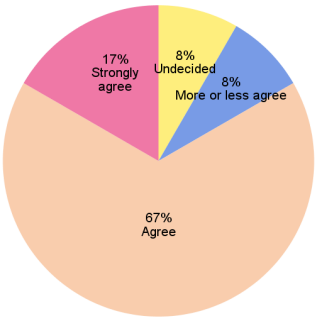


Figure 7.2: Distribution of the combined (median) scale for *Perceived usefulness*

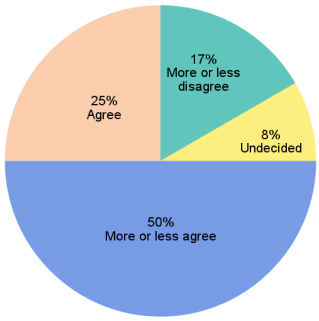


Figure 7.3: Distribution of responses to the question whether the framework helped the participant choosing a design (*perceived usefulness choice*)

Perceived usefulness We are able to combine multiple factors into the scale *perceived usefulness*, but have to analyze *perceived usefulness for choice* separately. The scale consists of the factors *perceived usefulness for context analysis, reflection, specification, translation, conflict detection, and auditing* (see 7.2.2). The scale is internally consistent (Cronbach's Alpha = 0.782), this is validated by the high correlation between the factors (see table C.1). We exclude factor *perceived usefulness choice*, as it drops Cronbach's alpha down to 0.684, which makes the scale doubtful. This is explained by the negative correlation with most other factors (table C.1), and the relatively low agreement with the statement (fig. 7.3).

Participants agree highly about their overall perception of usefulness, providing conclusive evidence of the usefulness of the prototype. We combined Likert scales using the median, as the mean is not a suitable measure of centrality for ordinal data. The result shows high agreement as 92% is positive about the usefulness of the prototype (fig. 7.2). The median is 6 ('agree') with an interquartile range of 0. The overall usefulness is reflected in comments from the participants, especially with respect to the specification phase. About the specification phase, participants stated *"I'm not really familiar with requirement engineering. But this tool helped me a lot and let me think about something that I might miss."*, *"Elicitation is very useful and seems like a brainstorm phase"*, *"Things that an experienced person wouldn't have thought of"*, *"Helps with understanding fairness if you're not familiar with it"*, *"Makes you think about fairness for individuals other than the applicant"*, and *"I think such a tool can be a tremendous help"*.

Despite the overall high perceived usefulness, the usefulness of the prototype for the design task declines over the course of the experiment. When investigating the response distributions to (figure 7.4), we observe that a large majority of the participants was satisfied by the performance of the prototype. However, a decline in perceived usefulness over the course of the performed design tasks is observed. Design tasks were performed in the order: *context analysis, reflection, specification, translation, choice, conflict detection.* This decline might be attributed to the fidelity of the prototype, as the user interface of the prototype was described as *"intimidating"*, *"elaborate"* and *"overwhelming"*.

The framework appears to be moderately successful in guiding the user towards a design. Although 75% is positive (fig. 7.3), the participants seem to agree much less to this statement than to those of other factors that comprise perceived usefulness. One participant stated that *"[t]he design guide helped more than the tool itself"*.

Conflict management The prototype provides satisfactory conflict *identification*, according to the majority of participants. 75% of participants agrees with the statement that the prototype helped them in identifying conflicts, half of which strongly agrees (see figure 7.5).

However, manual conflict identification might not scale very well beyond this experiment, as the amount of pairwise checks increases. On average participants performed 15 pairwise checks ($M = 2.5, SD = 0.7$) among which 2,5 conflicts are detected ($M = 2.5, SD = 0.7$). The highest number of

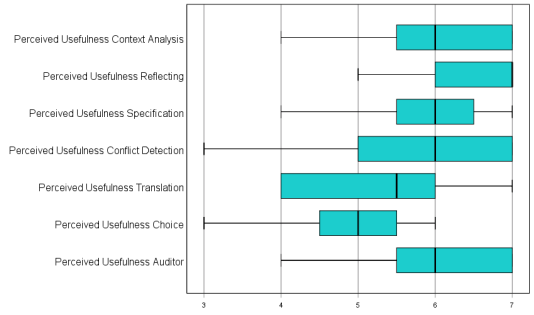


Figure 7.4: Boxplots of the response distributions for the factors that comprise the measure perceived usefulness. The plots have been sorted by the order in which the corresponding tasks were performed. A clear decline of usefulness is visible.

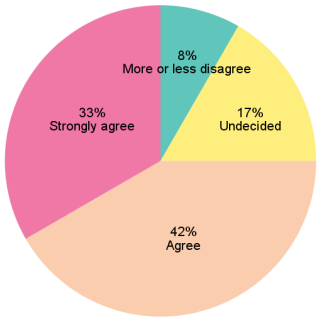


Figure 7.5: Distribution of responses to the question whether the framework helped the participant in identifying conflicts between requirements (*perceived usefulness conflict detection*)

checks is 36, from which 2 conflicts are identified. One participant noted this and suggested that if there are transitive relationships between requirements, this reduces the amount of checks.

Explication of design decisions and justifications Analyzing the quantitative data, we see that participants make many explicit design decisions during the use of the prototype. We observe that a small number of requirements leads to a large amount of explicit design decisions. On average 5 requirements of which 2 were pre-specified, leads to 17 explicit design decisions ($M = 17.3, SD = 2.6$) Mostly the requirement rationale and design solution, with on average 6 justifications ($M = 6.1, SD = 0.7$).

Second, we must analyze the factors that make up *explicitness* seperately, as they are not correlated enough to combine them into one scale. For one, the scale *explicitness* is not internally consistent (Cronbach's Alpha = 0.395), and the factors are not significantly correlated. This is confirmed by applying Principal Component Analysis, as the four factors each load on different components. For legibility, the figures are placed in appendix C.

A large majority responds positive to the statements that—during the design task—they made (and motivated) choices that they otherwise would not have made (or motivated). For *perceived explicitness*, 75% of participants was at least somewhat in agreement with the statement, while for *perceived justification* this was 83% (see figure C.2a and C.2b).

When it comes to auditing, almost all participants agreed that the designer made explicit assumptions, but are divided about understanding these assumptions. 92% of participants found that the designer was explicit about their fairness assumptions (fig. C.2c). There is no convincing evidence that participants understood assumptions, as only a slight majority agrees with the corresponding statement (fig. C.2d). This might be due to the limited number of justifications in the software design that was presented.

Traceability The prototype also provides satisfactory traceability, according to a large majority of participants. Nearly all participants (83%) respond positive to the statement that the prototype helped them in finding the value assumptions behind the software design during their audit task (see figure 7.6). The statement relates to their activity of backwards tracing from the software design to fairness requirements. Moreover, a third of all participants strongly agrees with the statement.

Perceived fairness Perceived fairness appears to be highly controversial. All participants were given the same document and task, but they are highly divided about their perception of fairness. A slight majority (58% is in agreement with the statement that they would consider a decision by the described system fair, even if it had a negative outcome for them (figure 7.7). This divergence is seen in comments from the participants. On one hand, a participant notes that *"[s]eeing that the tool has been used gives confidence in the completeness of the assessment"*. On the other hand, two participants remarked that they are not able to consider the software design fair, as it does not assure that the implementation will behave as what is described in the design.

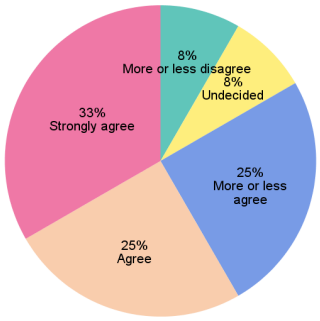


Figure 7.6: Distribution of responses to the question whether the framework helped the participant find the value assumptions behind the software design (*perceived traceability*)

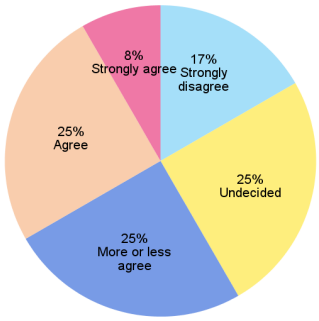


Figure 7.7: Distribution of responses to the question whether the participant would consider an automatic decision by the system fair, even if they get a negative outcome (*perceived fairness*)

Also, there is reason to believe that the audit view is useful for assessing the fairness of a software design. The majority of participants was able to assess the fairness of the software design within 15 minutes, as 75% of participants either finds the software design fair or not fair (fig. 7.7). A participant commented that the structure of the audit view, having all the assumptions visible, makes auditing easier. However, there is no significant correlation with the perceived usefulness of the prototype for their auditing task.

8

Conclusion

In this section, we discuss the results (section 8.1) and draw our final conclusion (section 8.2).

8.1. Discussion of the evaluation results

Our findings suggest that the Software Engineering for Values (SEfV) framework indeed supports the translation of the value fairness into a software design, and provides satisfactory support in explication and traceability. Participants found all but one feature helpful, and were very positive about the prototype's function of guiding in the reflection on fairness norms and specifying those into requirements. In fact, the fairness tree itself received far greater attention than expected. Elicitation by reflecting on this tree was found to be both structured as well as a brainstorm phase. The prompts that guide elicitation (fig. 6.6) actually functioned as 'cards' to elicit value judgments, similar to the function of *envisioning cards* (Friedman & Hendry, 2012). Furthermore, the SEfV framework also seems to support the auditing of design decisions and fairness assumptions.

Interestingly, we found an unexpected divergence in the perceptions of fairness among our participants. While our findings remain inconclusive as to whether applying the framework results in a fairer software design. However—in the controlled setting of the audit task—participants showed a great divergence in the perception of the systems fairness. This finding is in line with research on the perception of fairness in algorithmic decision-making (ADM). R. Wang et al. (2020) found the *development procedure* to have a moderating effect on the perceived fairness of algorithmic decision-making (R. Wang et al., 2020). Stronger factors are *algorithm outcome* A possible explanation, given by one participant, is the fact that the software design does not ensure that the implementation will behave accordingly. Future work may investigate which factors contribute to the fairness perception of auditors using the SEfV framework.

Limitations There are still challenges ahead for the application outside of our experiment. First of all, there are some limitations to conflict management within the prototype. Pairwise *conflict detection* was effective, but scales considerably with the amount of requirements that might be derived. This method might therefore not be feasible in a real-world setting. *Conflict resolution* must be evaluated in a real-world setting as it requires communication with stakeholders. In the experiments we therefore skipped this phase of SEfV.

Second, the prototype did not seem to help the participants in choosing design practices to satisfy the fairness requirements. This might be improved through an effort to make the design guide more extensive and usable. Within the scope of this study, a limited amount of knowledge on current practices has been combined. In a real-world setting, this design guide would be a knowledge base, updated over time. While there are many toolkits for debiasing (section 4.2.6), no 'open-source' knowledge base to guide the selection has been found.

Third, the user interface of the prototype may have inhibited the task performance. Usefulness ratings decline over the course of the design task, as participants were intimidated by the amount of information they were confronted with. This can be attributed to the fidelity of the user interface of the prototype, as for most tasks it provides all the information for that task in each worksheet. Given the

high scores given by participants, a higher fidelity—such as a web application—would be an interesting future direction.

8.2. Conclusion

In this study, we aimed to help practitioners design algorithmic decision-making software for fairness. We observed that the unexpected and biased behaviour that makes ADM systems problematic can be attributed to the lack of a method to specify what fairness means for the ADM system. To solve this problem, we set out to develop a framework—Software Engineering for Values (SEfV)—that would provide guidance in reflecting on values, help with specifying and translating requirements into the design, help choosing among conflicting solutions and assumptions, and explicating design decisions.

First, we reviewed fairness concepts in social sciences and conceptualized fairness (chapter 3). Second, we related the conceptualization to software design practices for fairness and identified gaps in the mapping (chapter 4). Next, we set out to design and develop the SEfV framework (chapter 5), demonstrate its application in a hypothetical design-for-fairness scenario (chapter 6), and evaluated its usefulness in user experiments (chapter 7).

Contribution 1: Fairness tree We conceptualized fairness based on literature in organizational justice (RQ 1.1). The resulting fairness tree consists of the value fairness specified into four dimensions (distributive, procedural, informational, interactional fairness)—further specified into 31 ADM-specific fairness norms (RQ 1.2). The conceptualization is supported by multiple documents in the review (1.2.1), and the organizational justice conceptualization has been used by other authors as well (Binns et al., 2018; German et al., 2018; Lee, Jain, et al., 2019; Robert et al., 2020).

We must realise, however, that the fairness theory from organizational justice is a descriptive, behavioral theory. This means that we should consider the conceptualization as descriptive and not normative (i.e. it should be suggestive and elicitive), and it may require extension in certain contexts.

Contribution 2: Mapping of design practices to aspects of fairness We then reviewed literature for fairness measures and interventions, categorized it by fairness assumption (e.g. fairness through non-discrimination) and mapped those categories to one or more of 31 norms in the fairness tree. We argue that this mapping is useful in two directions: 1) The mapping functions as a design guide when designing for norms; and 2) It functions as a framework that unifies the different fields of study (e.g. definitions of fairness with respect to stereotypes, problem formulation, non-discrimination).

Contribution 3: Research gaps in software design practice Furthermore, contribution 2 was accompanied by the identification of literal gaps in the mapping. In essence, those are research gaps that offer the fair ML community potential for cross-disciplinary research. These research gaps are analyzing and optimizing for consistency across time, optimizing for a target representation, less intrusive data collection, designing AI for mental and physical integrity.

Contribution 4: the Software Engineering for Values framework We developed the SEfV framework, using the Design Science Research Methodology. The framework had eight objectives, for which 19 requirements have been formulated (RQ 3.1). The design was elaborated using a structure implied by the functional requirements (RQ 3.2) and a prototype was made in Excel (RQ 3.3).

The resulting framework was applied to the illustrative case of a loan decisioning system that had to be designed for fairness (RQ 4.1). User experiments were able to confirm that the framework was perceived as sufficiently useful supported the activities (thus satisfying the criteria), except for the *choice* activity which we expect to be the result of the fidelity of the prototype (4.2).

Contribution 5: A general architecture for ADM systems From the application effort, we also contribute an architecture of ADM components and design parameters (4.1.1). The architecture is based on literature on Decision Support Systems and seminal works in decision-making theory.

8.2.1. Implications for researchers

This study aimed to bridge the gap between Software Engineering and Design for Values, which has been called for in multiple research roadmaps. Relating our work to the research roadmaps in Software

Engineering, we find that the SEfV framework is a contribution to the efforts of Mougouei et al. (2018), Aydemir and Dalpiaz (2018) and Brun and Meliou (2018). The SEfV framework also contributes to Design for Values as a new Engineering Design for Values method (van de Poel, 2015b).

The fairness tree seemed to function as a source for a cards-based requirement elicitation workshop such as Envisioning Cards (Friedman & Hendry, 2012).

8.2.2. Implications for practitioners

There is an increasing external pressure to design ethical AI-based systems. Within the next five to ten years, practitioners will—either for purposes of compliance, certification or corporate responsibility—have to demonstrably implement measures to safeguard ethical values like fairness, social good, human autonomy, etc.

Practitioners have the complicated task of integrating structural approach of Software Engineering with the analytical approach that is required in developing contingent ML-models for ADM systems.

The SEfV framework may be a first step towards a structural design method in Machine Learning systems that is both useful for the designer as auditor.

8.3. Future research directions

In this study, a conceptual framework for Software Engineering for Values (SEfV) is presented, but its use is only demonstrated for the value *fairness* and an *algorithmic decision-making* system. Future research directions are further inspection of the application in designing ADM for fairness, or extend the framework.

1. Investigate the application of the framework in a real-life context, team setting, using a higher fidelity artifact. The artifact should reduce information overload for the user as much as possible. E.g. a web application.
2. Investigating fairness perceptions of people who using the SEfV framework for auditing.
3. Developing an open source design guide for fairness design practices (as described in the discussion)
4. *fairness envisioning cards*, as described in the discussion

Extending the framework is possible along at least these three directions:

1. Design-for-x: Application the framework for a broader range of ethical values. The challenge is to incorporate the *discovery* activity (van de Poel, 2015b): the framework should support the discovery of values and their conceptualization.
2. Fairness measures and interventions: Where *bias correction* and *representation* are already being researched by the Fair-ML community, it might also be valuable to search and integrate ML measures and interventions to other fairness notions. For instance, *time consistency* is a fairness factor that contributes to procedural fairness, and is being researched within a different field as adversarial learning.
3. Software Engineering: The framework helps with the translation of needs to requirements, but future research is needed to verify that the ethical requirements have correctly been implemented and to validate that the system adheres to the fairness needs that.

A

Morphological chart

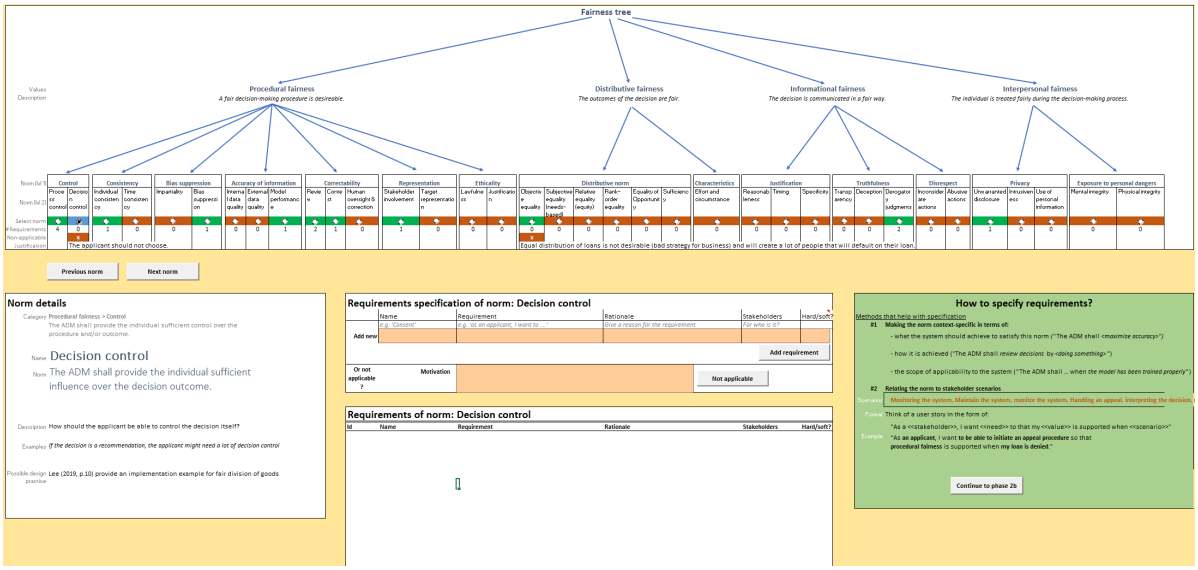
Table A.1: Morphological chart for the development of the SEfV framework. Selected alternatives are colored grey.

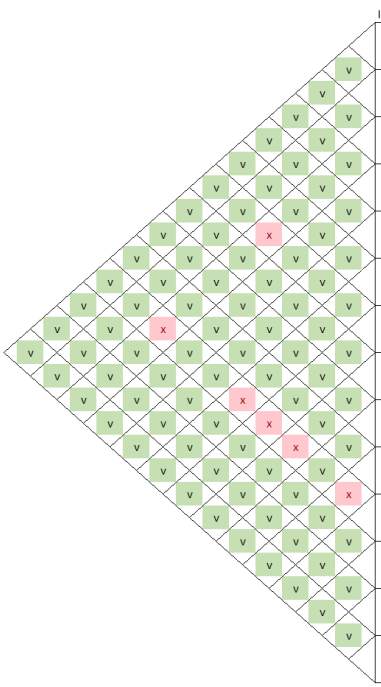
Objective	Design requirement		Means A	Means B	Means C
Agnosticism	A1	Requirements Engineering	[HTML]C0C0C0Elicitation, Analysis, Specification and Validation phases as described in Background chaptter		
	A2	Software Design	[HTML]C0C0C0Software design process as described in background		
Guidance	B1	Inform stages	[HTML]C0C0C0Introductory page that explains SEfV		
	B2	Inform action	[HTML]C0C0C0Info boxes		
	B3	Inform output	[HTML]C0C0C0Info boxes		
Specification	C1	Applicability	[HTML]C0C0C0While iterating over value concepts, prompt whether it is applicable	An applicability study stage early in the process, where the user marks which high-level norms are non-applicable.	The user can specify which requirements fall under Would have (meaning: non-applicable for now) of the MoSCoW model
	C2	Value specification	Domain analysis (Zowghi, 2005) from the point of view of the value conceptualization	Goal-based elicitation	Prompts based on Van de Poel (2013)
Translation	D1	Design parameters	[HTML]C0C0C0QFD relationship matrix	For each requirement, the user can select a design parameter where a solution is implemented to satisfy that requirement.	
Choice	E1	Incommensurability	[HTML]C0C0C0Add a classification category hard/soft	MoSCoW model: Hard requirements are grouped under Must haves of the MoSCoW model, soft requirements under Should have or Could have.	
	E2	Identify & analyze conflicts	Pairwise comparisons (Analytic Hierarchic Process)	Deontic logic & model checking (Esiava 2014)	
	E3	Resolve conflicts	top-10 reqs	numerical assignment (e.g. moscow model)	ranking (e.g. in QFD)
	E4	Feasibility	Respecification	innovation	Deontic logic & model checking (Esiava 2014)
	E5	Practice	[HTML]C0C0C0Produce a design guide that connects the design practices (e.g. Subgroup Fairness [Kearns et al 2018]) to fairness concepts	QFD Solution part	
Stakeholders	F1	Stakeholder analysis	[HTML]C0C0C0Value sensitive stakeholder analysis (Friedman et al 2008 ;Simons & Verhagen 2008)		
	F2	Stakeholder requirement elicitation	[HTML]C0C0C0Value stories	Adapting RE methods for value sensitive requirement elicitation (Interviews,	Value sensitive semi-structured interview (Friedman et al, 2008)
	F3	Stakeholder requirement tracing	[HTML]C0C0C0Requirements have an extra attribute: supporting stakeholders	Requirements grouped per stakeholder	
Negotiation	G1	Stakeholder involvement	[HTML]C0C0C0Conflicts between stakeholder requirements are highlighted and the designer is notified. The designer negotiates with stakeholders and resolves the conflict		
Explicitness	H1	Justification	[HTML]C0C0C0All design decisions are accompanied by a justification		
	H2	Evaluation	[HTML]C0C0C0One view that shows all design decisions at a high level, with underlying details in an accompanying document.	Nutritional label / model cards	

B

SEfV prototype

The prototype has been developed in Microsoft Excel. The following images exhibit the user interface.





Id	Name	Requirement	Type
3	**Training label check**	As a decision-maker, I want the training data not to contain pejorative labels (i.e. labels with a negative connotation).	Hard
4	**Explanation label check**	The system shall not use pejorative labels in its justification.	Hard
8	**Consent**	As an applicant, I want to be able to consent to the use of my information, so that my privacy is supported when I apply for a loan.	Hard
9	**Review**	Individuals shall be presented with what data points were actually used in the decision-making process and which priority they had	Hard
10	**Review action**	If the information is faulty, the individual is able to flag it.	Hard
11	**Process control**	As an applicant, I want to be able to select the data that are used to decide about my loan applications.	Hard
12	**Individual consistency**	Individuals with similar financial status should receive similar treatment.	Hard
13	**Unwanted Bias suppression**	The system should not use information unrelated to the loan decision for its classification, either directly or indirecly (in the form of a proxy).	Hard
14	**Cancellation**	As an applicant, I want to be able to cancel the application before the decision is made, without a negative result for me.	Hard
1	**Appeal procedure**	As an applicant, I want to be able to initiate an appeal procedure so that my procedural fairness is supported when I get an unfavourable decision.	Soft
2	**Credit risk model accuracy**	As a decision-maker, I want the accuracy of the credit risk scoring model to be as high as possible.	Soft
5	**Comments textbox**	As an applicant, I want to be able to voice my views before the loan application decision is being made.	Soft
6	**Comments check**	As an applicant, I want my views to be taken into account.	Soft
7	**Balanced team**	The development team consists of an equal representation of gender, race and age.	Soft

Design guide

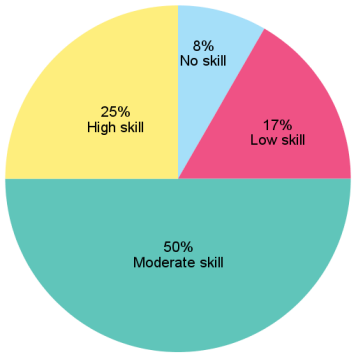
ID	Design requirements	ID for design guide	Solutions (What is done to satisfy this requirement?)	Justifications (Why this solution?)	Design parameter (Where is the solution implemented?)
1	Appeal procedure: As an applicant, I want to be able to initiate an appeal procedure so that my procedural fairness is supported when I get an unfavourable decision.	P.5.2	The subject UI contains an easily accessible form for appealing the loan decision.	By design, the applicant should be able to start the applea procedure from their main point of access (their user interface).	UI component: Subject UI
2	Credit risk model accuracy: As a decision-maker, I want the accuracy of the credit risk scoring model to be as high as possible.	P.4.3	During training, the model optimizes for accuracy.	Accuracy as a metric for performance reflects the bank's needs the best	Model component: Training strategy
3	Training label check: As a decision-maker, I want the training data not to contain pejorative labels (i.e. labels with a negative connotation).	IM.2.3	The system uses the German Credit Dataset.	The dataset has been checked againest labels that can be viewed as demeaning.	Database component: Data collection
4	Explanation label check: The system shall not use pejorative labels in its justification.	IM.2.3	Fair representation learning to remove words that have a negative connotation.	Fair representation learning is very effective in removing stereotypes from the data.	Model component: Training strategy
5	Comments textbox: As an applicant, I want to be able to voice my views before the loan application decision is being made.	P.1.1	The loan application form contains a textbox for comments.	As their point of access, the UI should allow the applicants to motivate their application.	UI component: Subject UI
6	Comments check: As an applicant, I want my views to be taken into account.	P.1.1	Comments are assessed by a loan officer, separate from the decision-making. If there is reason to override it, the loan officer is able to do so.		UI component: Decisionmaker UI
7	Balanced team: The development team consists of an equal representation of gender, race and age.	P.6.1	The IT team employs equal opportunity recruitment	Equal opportunity recruitment gives everyone the chance to get a job in this team.	Design phase: Team composition
8	Consent: As an applicant, I want to be able to consent to the use of my information, so that my privacy is supported when I apply for a loan.	IP.2.1	The loan application form contains a checkbox for approving the use of personal information for the decision.		UI component: Subject UI
9	Review: Individuals shall be presented with what data points were actually used in the decision-making process and which priority they had	P.5.1	The UI shows the applicant (also before the application) what information about them is available.	We don't want the applicant to change the data themselves.	UI component: Subject UI
10	Review action: If the information is faulty, the individual is able to flag it.	P.5.1	The application form tells the applicant what information they already have on the applicant, what information is necessary, how it will be used. If there is something wrong, the applicant can send a flag request.	We want to avoid the applicant unnecessarily flagging, creating more work for us.	UI component: Subject UI
11	Process control: As an applicant, I want to be able to select the data that are used to decide about my loan applications.	P.1.1	The subject UI should provide the user with the option to select the data sources that should be taken into account during the evaluation process.	By design, the application should be able to control the information that they provide (in the user interface)	UI component: Subject UI
12	Individual consistency: Individuals with similar financial status should receive similar treatment.	P.2.1	During training, the model should optimize its performance while treating subjects with similar fincancial status in the same manner.	Accuracy needs to be optimized but constrained for indiviual fair treatment.	Model component: Data imbalance correction
13	Unwanted Bias suppression: The system should not use information unrelated to the loan decision for its classification, either directly or indirecly (in the form of a proxy).	P.3.2	During training and infrence, the model should not use protected attributes either directly or indirectly for its decisions	Performance should remain the same regardless of the protected attributes values	Model component: Data imbalance correction
14	As an applicant, I want to be able to cancel the application before the decision is made, without a negative result for me.	P.1.1			UI component: Subject UI

C

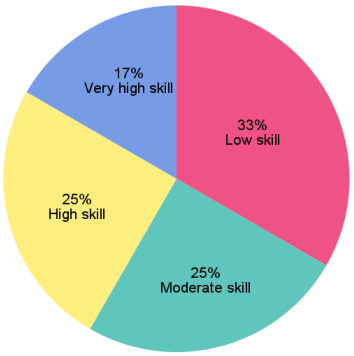
Detailed evaluation results

In this appendix, the following details results are found:

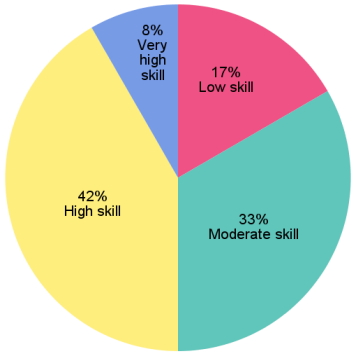
- Frequency distributions for the self-reported skills of the participants
- Frequency distributions for the factors comprising evaluation criterion *Explication of design decisions*
- Correlation matrix for the factors comprising evaluation criterion *Perceived Usefulness*



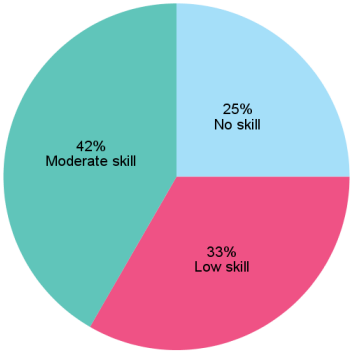
(a) Self-reported skills in Requirements Engineering



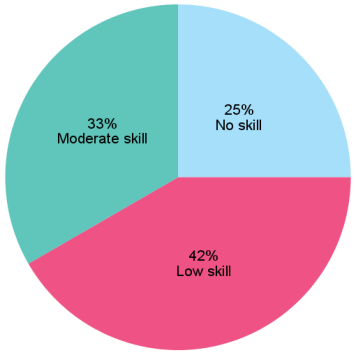
(b) Self-reported skills in Software Design



(c) Self-reported skills in Machine Learning

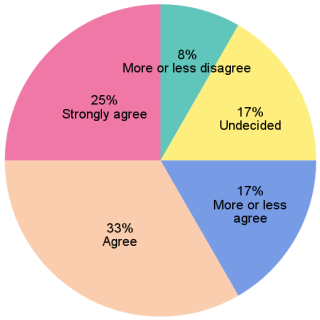


(d) Self-reported skills in Fairness in Machine Learning (debiasing etc.)

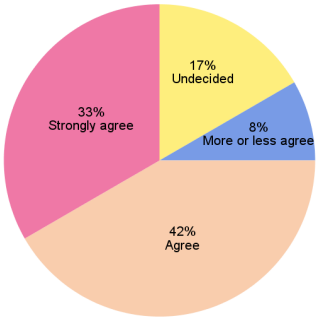


(e) Self-reported skills in Design for Values

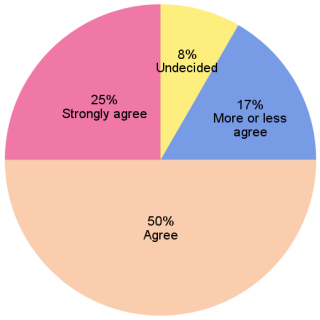
Figure C.1: Distribution of self-reported skills over the participants



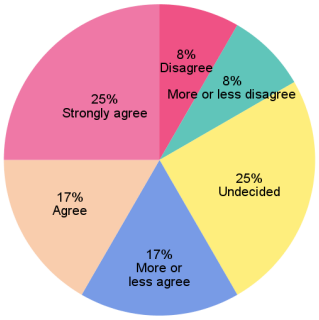
(a) Distribution of responses to the question whether the participant made choices that they otherwise would have made implicitly (*perceived explicitness as designer*)



(b) Distribution of responses to the question whether the participant motivated choices that they otherwise would not have motivated (*perceived justification as designer*)



(c) Distribution of responses to the question whether the participant found that the designer made explicit assumptions about what they consider fair (*perceived explicitness as auditor*)



(d) Distribution of responses to the question whether the participant understood why the designer made these assumptions (*perceived justification as auditor*)

Figure C.2: Distribution of responses to concepts that comprise the criterion *explicitness*

Table C.1: Spearman rank correlations for the factors comprising evaluation criterion *explicitness*

	Perceived Usefulness Designer	Perceived Usefulness Context Analysis	Perceived Usefulness Reflecting	Perceived Usefulness Specification	Perceived Usefulness Translation	Perceived Usefulness Choice	Perceived Usefulness Conflict Detection	Perceived Usefulness Auditor	Perceived Usefulness (Combined median scale)	Perceived Usefulness (Mean)
Perceived Usefulness Designer	1,000	,621*	0,454	0,337	0,187	-0,135	0,139	,591*	0,557	,683*
Perceived Usefulness Context Analysis	,621*	1,000	0,312	0,028	0,039	-0,431	,686*	,765**	,707*	,756**
Perceived Usefulness Reflecting	0,454	0,312	1,000	0,236	0,410	-0,082	0,217	-0,097	0,288	,648*
Perceived Usefulness Specification	0,337	0,028	0,236	1,000	,809**	0,096	0,112	0,212	0,295	0,542
Perceived Usefulness Translation	0,187	0,039	0,410	,809**	1,000	0,315	0,116	-0,075	0,300	0,542
Perceived Usefulness Choice	-0,135	-0,431	-0,082	0,096	0,315	1,000	-0,557	-0,377	-0,340	-0,380
Perceived Usefulness Conflict Detection	0,139	,686*	0,217	0,112	0,116	-0,557	1,000	0,531	,780**	,603*
Perceived Usefulness Auditor	,591*	,765**	-0,097	0,212	-0,075	-0,377	0,531	1,000	,691*	0,516
Perceived Usefulness (Combined median scale)	0,557	,707*	0,288	0,295	0,300	-0,340	,780**	,691*	1,000	,724**
Perceived Usefulness (Mean)	,683*	,756**	,648*	0,542	0,542	-0,380	,603*	0,516	,724**	1,000

*. Correlation is significant at the 0.05 level (2-tailed).
**. Correlation is significant at the 0.01 level (2-tailed).

Bibliography

Adams, J. S. (1965). Inequity In Social Exchange. *Advances in Experimental Social Psychology*, *2*(100), 267–299. https://doi.org/10.1016/S0065-2601(08)60108-2

Aizenberg, E., & van den Hoven, J. (2020). *Designing for Human Rights in AI*, Delft University of Technology.

Aldewereld, H., Dignum, V., & Tan, Y. h. (2015). Design for values in software development (J. van den Hoven, P. E. Vermaas, & I. van de Poel, Eds.). In J. van den Hoven, P. E. Vermaas, & I. van de Poel (Eds.), *Handbook of ethics, values, and technological design: Sources, theory, values and application domains*. Dordrecht, Springer Netherlands. https://doi.org/10.1007/978-94-007-6970-0

Almada, M. (2019). Human intervention in automated decision-making: Toward the construction of contestable systems, In *International conference on artificial intelligence and law (icail '19)*, Montreal, QC, Canada. https://doi.org/10.13140/RG.2.2.19766.55368/1

Anton, R. J. (1990). Drawing the line: An exploratory test of ethical behavior in negotiations. https://doi.org/10.1108/eb022683

Arnott, D. (1998). A Taxonomy of Decision Biases. *School of Information Management & Systems*, 1–48. http://www.sims.monash.edu.au/staff/darnott/biastax.pdf

Aurum, A., & Wohlin, C. (Eds.). (2005). *Engineering and Managing Software Requirements*. Springer.

Aydemir, F. B., & Dalpiaz, F. (2018). A roadmap for ethics-aware software engineering, In *Proceedings - international conference on software engineering*, New York, New York, USA, ACM Press. https://doi.org/10.1145/3194770.3194778S

Balayn, A., Lofi, C., & Houben, G.-J. (2020). *Controlling Biases in the DBMS : Fairness by Design for Data-Driven Decision Support Systems [Vision]*, Delft University of Technology.

Barn, B., Barn, R., & Raimondi, F. (2015). On the Role of Value Sensitive Concerns in Software Engineering Practice. *Proceedings - International Conference on Software Engineering*, *2*, 497–500. https://doi.org/10.1109/ICSE.2015.182

Barocas, S., Crawford, K., Shapiro, A., & Wallach, H. (2017). The problem with bias: from allocative to representational harms in machine learning, In *Special interest group for computing, information and society*.

Barocas, S., Hardt, M., & Narayanan, A. (2017). Fairness in Machine Learning, In *Nips tutorial*. https://fairmlbook.org/pdf/fairmlbook.pdf

Barocas, S., & Selbst, A. D. (2016). Big Data's Disparate Impact. *California Law Review*, *104*(671).

Ben-Porat, O., Sandomirskiy, F., & Tennenholtz, M. (2019). Protecting the Protected Group: Circumventing Harmful Fairness, arXiv 1905.10546, 1–24. http://arxiv.org/abs/1905.10546

Berander, P., & Andrews, A. (2005). Requirements Prioritization, In *Engineering and managing software requirements*.

Berk, R., Heidari, H., Jabbari, S., Kearns, M., & Roth, A. (2018). Fairness in Criminal Justice Risk Assessments: The State of the Art. *Sociological Methods and Research*, arXiv 1703.09207, 1–42. https://doi.org/10.1177/0049124118782533

Bies, R., & Moag, J. (1986). *Interactional Justice: Communication Criteria of Fairness* (R. Lewicki, B. Sheppard, & M. Bazerman, Eds.; Vol. 1). Greenwich.

Bies, R. J. (2015). Interactional Justice: Looking Backward, Looking Forward (R. S. Cropanzano & M. L. Ambrose, Eds.). In R. S. Cropanzano & M. L. Ambrose (Eds.), *The Oxford Handbook of Justice in the Workplace*. Oxford, Oxford University Press. https://doi.org/10.1093/oxfordhb/9780199981410.001.0001

Bies, R. J., & Greenberg, J. (2017). Justice, Culture, and Corporate Image: The Swoosh, the Sweatshops, and the Sway of Public Opinion, In *The blackwell handbook of cross-cultural management*. Blackwell Publishing Ltd. https://doi.org/10.1002/9781405164030.ch16

Bies, R. J., & Tripp, T. M. (1993). Employee-initiated defamation lawsuits: Organizational responses and dilemmas. *Employee Responsibilities and Rights Journal*, *6*(4), 313–324. https://doi.org/10.1007/BF01385020

Binns, R. (2017). Fairness in Machine Learning: Lessons from Political Philosophy, arXiv 1712.03586. http://arxiv.org/abs/1712.03586

Binns, R., Van Kleek, M., Veale, M., Lyngs, U., Zhao, J., & Shadbolt, N. (2018). 'It's reducing a human being to a percentage'; perceptions of justice in algorithmic decisions. *Conference on Human Factors in Computing Systems - Proceedings*, *2018-April*arXiv arXiv:1801.10408v1. https://doi.org/10.1145/3173574.3173951

Blodgett, S. L., Barocas, S., Daumé, H., & Wallach, H. (2020). Language (Technology) is Power: A Critical Survey of "Bias"in NLP, (100), arXiv 2005.14050. http://arxiv.org/abs/2005.14050

Bolukbasi, T., Chang, K.-W., Zou, J., Saligrama, V., & Kalai, A. (2016). Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embedding. *30th Conference on Neural Information Processing Systems*, (NIPS 2016), arXiv 1607.06520, 1–9. https://code.google.com/archive/p/word2vec/

Brun, Y., & Meliou, A. (2018). Software fairness. *ESEC/FSE 2018 - Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 754–759. https://doi.org/10.1145/3236024.3264838

Bublitz, C. (2013). My Mind Is Mine!? Cognitive Liberty as a Legal Concept (A. G. Franke & E. Hildt, Eds.). In A. G. Franke & E. Hildt (Eds.), *Cognitive enhancement: An interdisciplinary perspective.* Springer. https://doi.org/10.1007/978-94-007-6253-4

Buolamwini, J., & Gebru, T. (2018). Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification, In *Proceedings of the 1st conference on fairness, accountability and transparency*.

Chen, J., Kallus, N., Mao, X., Svacha, G., & Udell, M. (2019). Fairness under unawareness: Assessing disparity when protected class is unobserved, In *Fat* 2019 - proceedings of the 2019 conference on fairness, accountability, and transparency*. https://doi.org/10.1145/3287560.3287594

Chiheb, F., Boumahdi, F., & Bouarfa, H. (2019). A new model for integrating big data into phases of decision-making process. *Procedia Computer Science*, *151*(2018), 636–642. https://doi.org/10.1016/j.procs.2019.04.085

Child Rights International Network. (n.d.). Bodily integrity. Retrieved July 21, 2020, from https://archive.crin.org/en/home/what-we-do/policy/bodily-integrity.html

Citron, D. K. (2008). Technological Due Process, *85*(6).

Colquitt, J. A. (2001). On the dimensionality of organizational justice: A construct validation of a measure. *Journal of Applied Psychology*, *86*(3), 386–400. https://doi.org/10.1037/0021-9010.86.3.386

Colquitt, J. A., & Rodell, J. B. (2015). Measuring Justice and Fairness (R. S. Cropanzano & M. L. Ambrose, Eds.). In R. S. Cropanzano & M. L. Ambrose (Eds.), *The Oxford Handbook of Justice in the Workplace*. Oxford, Oxford University Press. https://doi.org/10.1093/oxfordhb/9780199981410.001.0001

Colquitt, J. A., Wesson, M. J., Porter, C. O., Conlon, D. E., & Ng, K. Y. (2001). Justice at the millennium: A meta-analytic review of 25 years of organizational justice research. *Journal of Applied Psychology*, *86*(3), 425–445. https://doi.org/10.1037/0021-9010.86.3.425

Cook, K. S., & Hegtvedt, K. A. (1983). Distributive Justice, Equity, and Equality. *Annual Reviews of Sociology*, *9*, 217–241.

Corbett-Davies, S., Pierson, E., Feller, A., Goel, S., & Huq, A. (2017). Algorithmic decision making and the cost of fairness. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *Part F1296*arXiv 1701.08230, 797–806. https://doi.org/10.1145/3097983.3098095

About alll the trade-offs regarding fairness

Council, C. (1999). Federal enterprise architecture framework (feaf). *United States Office of Management and Budget (September 1999)*.

Craig, J. N. (2016). Incarceration, Direct Brain Intervention, and the Right to Mental Integrity – a Reply to Thomas Douglas. *Neuroethics*, *9*(2), 107–118. https://doi.org/10.1007/s12152-016-9255-x

Definition of mental integrity

Crawford, K., & Calo, R. (2016). There is a blind spot in AI research. Nature Publishing Group. https://doi.org/10.1038/538311a

Cropanzano, R., Rupp, D. E., Mohler, C. J., & Schminke, M. (2001). Three roads to organizational justice. *Research in Personnel and Human Resources Management*, *20*, 1–113. https://doi.org/10.1016/S0742-7301(01)20001-2

De-Arteaga, M., Fogliato, R., & Chouldechova, A. (2020). A Case for Humans-in-the-Loop: Decisions in the Presence of Erroneous Algorithmic Scores, arXiv 2002.08035, 1–12. https://doi.org/10.1145/3313831.3376638

Deldjoo, Y., Anelli, V. W., Zamani, H., Bellogín, A., & Noia, T. D. (2019). Recommender systems fairness evaluation via generalized cross entropy, In *Ceur workshop proceedings*. https://arxiv.org/abs/1908.06708%20http://arxiv.org/abs/1908.06708

Detweiler, C. A., Hindriks, K. V., & Jonker, C. M. (2010). Principles for Value-Sensitive Agent-Oriented Software Engineering (D. Weyns & M.-P. Gleizes, Eds.). In D. Weyns & M.-P. Gleizes (Eds.), *Agent-oriented software engineering xi*. Toronto, CA, Springer.

Detweiler, C., & Harbers, M. (2014). Value stories: Putting human values into requirements engineering. *CEUR Workshop Proceedings*, *1138*, 2–11.

Deutsch, M. (1975). Equity, Equality, and Need: What Determines Which Value Will Be Used as the Basis of Distributive Justice? *Journal of Social Issues*, *31*(3), 137–149. https://doi.org/10.1111/j.1540-4560.1975.tb01000.x

Dose, J. J. (1997). Work values: An integrative framework and illustrative application to organizational socialization. *Journal of Occupational and Organizational Psychology*, *70*(3), 219–240. https://doi.org/10.1111/j.2044-8325.1997.tb00645.x

Dunkelau, J., & Leuschel, M. (2020). Fairness-Aware Machine Learning: An Extensive Overview, 1–60.

Dwork, C., Hardt, M., Pitassi, T., Reingold, O., & Zemel, R. (2012). Fairness through awareness, In *Itcs 2012 - innovations in theoretical computer science conference*. https://doi.org/10.1145/2090236.2090255

Dyer, R. (1997). *White* (1st). Routledge. https://www.oxfordreference.com/view/10.1093/oi/authority.20110803095804630

Dym, C. L. (1994). *Engineering Design: A Synthesis of Views* (1st). Cambridge, Cambridge University Press.

Eckoff, T. (1974). *Justice: Its Determinants in Social Interaction*. Rotterdam, NL, Rotterdam University Press. https://psycnet.apa.org/record/1975-07288-000

Esiava, S. (2014). *Harmonizing Norms with a Software Design* (Doctoral dissertation). Delft University of Technology.

European Union. (2016a). Charter of Fundamental Rights of the European Union. https://doi.org/10.4067/S0718-52002018000100167

European Union. (2016b). Regulation 2016/679 of the European parliament and the Council of the European Union. *Official Journal of the European Communities*, *L119*, 1–88.

Finegan, J. E. (2000). The impact of person and organizational values on organizational commitment. *Journal of Occupational and Organizational Psychology*, *73*(2), 149–169. https://doi.org/10.1348/096317900166958

Folger, R., & Cropanzano, R. (1998). *Organizational Justice and Human Resource Management*. SAGE. https://doi.org/10.4135/9781452225777

Folger, R., & Cropanzano, R. (2001). Fairness Theory: Justice as Accountability (R. Folger & J. Greenberg, Eds.). In R. Folger & J. Greenberg (Eds.), *Advances in organizational justice*. Stanford, CA, Stanford University Press.

Frankfurt, H. G. (2015). *On inequality*. Princeton, NJ, Princeton University Press. https://doi.org/10.2478/disp-2016-0008

Friedman, B., Kahn Jr, P. H., & Borning, A. (2008). Value Sensitive Design and information systems, In *Human-computer interaction in management information systems: Foundations*. NY, M.E. Sharpe.

Friedman, B., & Hendry, D. (2019). *Value sensitive design: shaping technology with moral imagination*. Cambridge, MA, MIT Press. https://books.google.com/books?hl=nl%7B%5C&%7Dlr=%7B%5C&%7Did=8ZiWDwAAQBAJ%7B%5C&%7Doi=fnd%7B%5C&%7Dpg=PR13%7B%5C&%7Ddq=value+sensitive+design+friedman+hendry%7B%5C&%

7Dots=vbliCwKuOR%7B%5C&%7Dsig=6e6ilxMKuOSN2OaPVfYzec0vqA4%20https://mitpress.mit.edu/books/value-sensitive-design

Friedman, B., & Hendry, D. G. (2012). The Envisioning Cards: A toolkit for catalyzing humanistic and technical imaginations. *Conference on Human Factors in Computing Systems - Proceedings*, 1145–1148. https://doi.org/10.1145/2207676.2208562

Friedman, B., Kahn Jr, P. H., & Borning, A. (1987). Value Sensitive Design: Theory and Methods. *Science and Technology Libraries*, *7*(3), 29–40. https://doi.org/10.1080/122v07n03_04

Friedman, B., & Nissenbaum, H. (1996). Bias in Computer Systems. *ACM Transactions on Information Systems*, *14*(3), 330–347. https://doi.org/10.1145/230538.230561

Galhotra, S., Brun, Y., & Meliou, A. (2017). Fairness testing: Testing software for discrimination. *Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, *Part F1301*arXiv 1709.03221, 498–510. https://doi.org/10.1145/3106237.3106277

Gebru, T., Morgenstern, J., Vecchione, B., Vaughan, J. W., Wallach, H., Daumé, H., & Crawford, K. (2018). Datasheets for Datasets, arXiv 1803.09010. http://arxiv.org/abs/1803.09010

German, D. M., Robles, G., Poo-Caamaño, G., Yang, X., Iida, H., & Inoue, K. (2018). Was my contribution fairly reviewed?: a framework to study the perception of fairness in modern code reviews. *Proceedings of the 40th International Conference on Software Engineering*, (2), 523–534. https://doi.org/10.1145/3180155.3180217

Grgić-Hlača, N., Zafar, M. B., Gummadi, K. P., & Weller, A. (2018). Beyond distributive fairness in algorithmic decision making: Feature selection for procedurally fair learning, In *32nd aaai conference on artificial intelligence, aaai 2018*.

Gummadi, K. P., & Heidari, H. (2019). Economic theories of distributive justice for fair machine learning. *The Web Conference 2019 - Companion of the World Wide Web Conference, WWW 2019*, 1301–1302. https://doi.org/10.1145/3308560.3320101

Harbers, M., Detweiler, C., & Neerincx, M. A. (2015). Embedding stakeholder values in the requirements engineering process, In *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)*, Springer, Cham. https://doi.org/10.1007/978-3-319-16101-3_23

Hardt, M., Price, E., & Srebro, N. (2016). *Equality of Opportunity in Supervised Learning*. https://doi.org/10.1109/ICCV.2015.169

Harris, J. G., & Davenport, T. H. (2005). Automated decision making comes of age. *MIT Sloan Management Review*, 2–10.

Hartley, J. (2003). *Communication, Cultural and Media Studies: The Key Concepts* (3rd ed.). Routledge.

Hauser, J. R., Griffin, A., Klein, R. L., Katz, G. M., & Gaskin, S. P. (2010). Quality Function Deployment (QFD), In *Wiley international encyclopedia of marketing*. Chichester, UK, John Wiley & Sons, Ltd. https://doi.org/10.1002/9781444316568.wiem05023

Heidari, H., Gummadi, K. P., Ferrari, C., & Krause, A. (2018). Fairness behind a veil of ignorance: A welfare analysis for automated decision making, In *Advances in neural information processing systems*. http://papers.nips.cc/paper/7402-fairness-behind-a-veil-of-ignorance-a-welfare-analysis-for-automated-decision-making

Heidari, H., Gummadi, K. P., Loi, M., & Krause, A. (2019). A moral framework for understanding fair ML through economic models of equality of opportunity, In *Fat* 2019 - proceedings of the 2019 conference on fairness, accountability, and transparency*, New York, New York, USA, ACM Press. https://doi.org/10.1145/3287560.3287584

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, *28*(1), 75–105. http://www.springerlink.com/index/pdf/10.1007/s11576-006-0028-8

High-Level Expert Group on Artificial Intelligence. (2019). Ethics Guidelines for Trustworthy AI, 2–36. https://ec.europa.eu/futurium/en/ai-alliance-consultation/guidelines%7B%5C#%7DTop

Holland, S., Hosny, A., Newman, S., Joseph, J., & Chmielinski, K. (2018). The Dataset Nutrition Label: A Framework To Drive Higher Data Quality Standards, (May), arXiv 1805.03677. http://arxiv.org/abs/1805.03677

Holstein, K., Vaughan, J. W., Daumé, H., Dudík, M., & Wallach, H. (2019). Improving fairness in machine learning systems: What do industry practitioners need?, In *Conference on human factors in computing systems - proceedings*. https://doi.org/10.1145/3290605.3300830

Homans, G. C. (1961). *Social Behavior: Its Elementary Forms* (1st). New York, NY, Harcourt, Brace & World. https://doi.org/10.2307/587952

Hossain, S., Mladenovic, A., & Shah, N. (2019). Designing Fairly Fair Classifiers Via Economic Fairness Notions, In *Www 2020: International world wide web conference*.

IEEE Computer Society. (2014). *Guide to the Software Engineering Body of Knowledge (SWEBOK)* (P. Bourque & R. E. Fairley, Eds.; Version 3.). www.swebok.org

Joseph, M., Kearns, M., Morgenstern, J., Neel, S., & Roth, A. (2016). Fair Algorithms for Infinite and Contextual Bandits, arXiv 1610.09559, 1–26. http://arxiv.org/abs/1610.09559

Kahn, W. A. (1990). Psychological conditions of personal engagement and disengagement at work. *Academy of Management Journal*, *33*(4), 692–724. https://doi.org/10.5465/256287

Karanasiou, A. P., & Pinotsis, D. A. (2017). A study into the layers of automated decision-making: emergent normative and legal aspects of deep learning. *International Review of Law, Computers and Technology*, *31*(2), 170–187. https://doi.org/10.1080/13600869.2017.1298499

Kärkkäinen, K., & Joo, J. (2020). FairFace: A Novel Face Attribute Dataset for Bias Measurement and Mitigation.

Kay, M., Matuszek, C., & Munson, S. A. (2015). Unequal representation and gender stereotypes in image search results for occupations. *Conference on Human Factors in Computing Systems - Proceedings*, *2015-April*, 3819–3828. https://doi.org/10.1145/2702123.2702520

Kearns, M., Neel, S., Roth, A., & Wu, Z. S. (2018). Preventing fairness gerrymandering: Auditing and learning for subgroup fairness, In *35th international conference on machine learning, icml 2018*. https://www.semanticscholar.org/paper/Preventing-Fairness-Gerrymandering%7B%5C%%7D3A-Auditing-and-Kearns-Neel/74289572067a8ba3dbe1abf84d4a352b8bb4740f

Koen, S. (2019). Not yet another article on Machine Learning! Retrieved August 18, 2020, from https://towardsdatascience.com/not-yet-another-article-on-machine-learning-e67f8812ba86

Kroes, P., & van de Poel, I. (2015). Design for Values and the Definition, Specification, and Operationalization of Values (J. van den Hoven, P. E. Vermaas, & I. van de Poel, Eds.). In J. van den Hoven, P. E. Vermaas, & I. van de Poel (Eds.), *Handbook of ethics, values, and technological design: Sources, theory, values and application domains*. Dordrecht, Springer Netherlands. https://doi.org/10.1007/978-94-007-6970-0

Kuhlman, C., Jackson, L., & Chunara, R. (2020). No computation without representation: Avoiding data and algorithm biases through diversity, arXiv 2002.11836. http://arxiv.org/abs/2002.11836

Kusner, M., Loftus, J., Russell, C., & Silva, R. (2017). Counterfactual fairness. *Advances in Neural Information Processing Systems*, *2017-Decem*(Nips), arXiv 1703.06856, 4067–4077.

Lahoti, P., Gummadi, K. P., & Weikum, G. (2019). Operationalizing individual fairness with pairwise fair representations, In *Proceedings of the vldb endowment*. https://doi.org/10.14778/3372716.3372723

Lee, M. K., Jain, A., Cha, H. J., Ojha, S., & Kusbit, D. (2019). Procedural justice in algorithmic fairness: Leveraging transparency and outcome control for fair algorithmic mediation. *Proceedings of the ACM on Human-Computer Interaction*, *3*(CSCW). https://doi.org/10.1145/3359284

Lee, M. K., Kahng, A., Kim, J. T., Yuan, X., Chan, A., Lee, S., Procaccia, A. D., Kusbit, D., See, D., Nooth-Igattu, R., & Psomas, A. (2019). WeBuildAI: Participatory Framework for Algorithmic Governance. *Proc. ACM Hum.-Comput. Interact*, *3*(November). https://doi.org/10.1145/3359283

Lepri, B., Oliver, N., Letouzé, E., Pentland, A., & Vinck, P. (2018). Fair, Transparent, and Accountable Algorithmic Decision-making Processes: The Premise, the Proposed Solutions, and the Open Challenges. *Philosophy and Technology*, *31*(4), 611–627. https://doi.org/10.1007/s13347-017-0279-x

Leventhal, G. S. (1976). The Distribution of Rewards and Resources in Groups and Organizations. *Advances in Experimental Social Psychology*, *9*(100), 91–131. https://doi.org/10.1016/S0065-2601(08)60059-3

Leventhal, G. S. (1980). What Should Be Done with Equity. *Social Exchange*.

Lewis, P. J. (1991). The decision making basis for information systems: the contribution of Vickers' concept of appreciation to a soft systems perspective. *European Journal of Information Systems*, *1*(1), 33–44. https://doi.org/10.1057/ejis.1991.5

Lind, E. A., & Tyler, T. R. (1988). *The social psychology of procedural justice. Critical issues in social justice*.

Liu, X. F. (2000). Software quality [function] deployment. *IEEE Potentials*, 14–16.

Loi, M., Herlitz, A., & Heidari, H. (2019). A Philosophical Theory of Fairness for Prediction-Based Decisions. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3450300

Madras, D., Pitassi, T., & Zemel, R. (2018). Predict responsibly: Improving fairness and accuracy by learning to defer. *Advances in Neural Information Processing Systems*, *2018-Decem*(NeurIPS), arXiv 1711.06664, 6147–6157.

Mantelero, A. (2018). AI and Big Data: A blueprint for a human rights, social and ethical impact assessment. *Computer Law and Security Review*, *34*(4), 754–772. https://doi.org/10.1016/j.clsr.2018.05.017

Martin, D., Prabhakaran, V., Kuhlberg, J., Smart, A., & Isaac, W. S. (2020). Participatory Problem Formulation for Fairer Machine Learning Through Community Based System Dynamics, arXiv 2005.07572, 1–6. http://arxiv.org/abs/2005.07572

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., & Galstyan, A. (2019). A Survey on Bias and Fairness in Machine Learning, arXiv 1908.09635. http://arxiv.org/abs/1908.09635

Miceli, M., & Castelfranchi, C. (1989). A Cognitive Approach to Values. *Journal for the Theory of Social Behaviour*, *19*(2), 169–193. https://doi.org/10.1111/j.1468-5914.1989.tb00143.x

Mitchell, M., Wu, S., Zaldivar, A., Barnes, P., Vasserman, L., Hutchinson, B., Spitzer, E., Raji, I. D., & Gebru, T. (2019). Model cards for model reporting, In *Fat* 2019 - proceedings of the 2019 conference on fairness, accountability, and transparency*. https://doi.org/10.1145/3287560.3287596

Molnar, P., & Gill, L. (n.d.). Bots at the gate: A human rights analysis of automated decision-making in canada's immigration and refugee system. *2018*
Simple explanation of concepts + good examples In-depth account of fairness in a complete case.

Monteith, S., & Glenn, T. (2016). Automated Decision-Making and Big Data: Concerns for People With Mental Illness. *Current Psychiatry Reports*, *18*(12). https://doi.org/10.1007/s11920-016-0746-6

Mougouei, D., Perera, H., Hussain, W., Shams, R., & Whittle, J. (2018). Operationalizing human values in software: A research roadmap. *ESEC/FSE 2018 - Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 780–784. https://doi.org/10.1145/3236024.3264843

Mozannar, H., & Sontag, D. (2020). Consistent Estimators for Learning to Defer to an Expert, arXiv 2006.01862. http://arxiv.org/abs/2006.01862

Obermeyer, Z., Powers, B., Vogeli, C., & Mullainathan, S. (2019). Dissecting racial bias in an algorithm used to manage the health of populations. *Science*, *366*(6464), 447–453. https://doi.org/10.1126/science.aax2342

Olteanu, A., Castillo, C., Diaz, F., & Kıcıman, E. (2019). Social Data: Biases, Methodological Pitfalls, and Ethical Boundaries. *Frontiers in Big Data*, *2*, 1–44. https://doi.org/10.3389/fdata.2019.00013

Passi, S., & Barocas, S. (2019). Problem formulation and fairness, In *Fat* 2019 - proceedings of the 2019 conference on fairness, accountability, and transparency*, New York, New York, USA, ACM Press. https://doi.org/10.1145/3287560.3287567

Peffers, K., Rothenberger, M., Tuunanen, T., & Vaezi, R. (2012). Design science research evaluation, In *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)*. https://doi.org/10.1007/978-3-642-29863-9_29

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, *24*(3), 45–77. https://doi.org/10.2753/MIS0742-1222240302

Pohl, K., & Rupp, C. (2015). *Requirement Engineering Fundamentals* (2nd). Santa Barbara, CA, Rocky Nook. https://doi.org/10.7312/mill15040-006

Power, D. J. (2002). *Decision Support Systems: Concepts and Resources for Managers* (Vol. 20).

Prat, N., Comyn-Wattiau, I., & Akoka, J. (2015). A Taxonomy of Evaluation Methods for Information Systems Artifacts. *Journal of Management Information Systems*, *32*(3), 229–267. https://doi.org/10.1080/07421222.2015.1099390

Rahwan, I. (2018). Society-in-the-loop: programming the algorithmic social contract. *Ethics and Information Technology*, *20*(1), arXiv 1707.07232, 5–14. https://doi.org/10.1007/s10676-017-9430-8

Reisman, D., Schultz, J., Crawford, K., & Whittaker, M. (2018). Algorithmic impact assessments: A practical framework for public agency accountability. *AI Now Institute*, (April), 22. https://ainowinstitute.org/aiareport2018.pdf

Robert, L. P., Pierce, C., Marquis, L., Kim, S., & Alahmad, R. (2020). Designing fair AI for managing employees in organizations: a review, critique, and design agenda. *Human-Computer Interaction*. https://doi.org/10.1080/07370024.2020.1735391

Ruf, B., Boutharouite, C., & Detyniecki, M. (2020). Getting Fairness Right: Towards a Toolbox for Practitioners, arXiv 2003.06920. http://arxiv.org/abs/2003.06920

Rumbaugh, J., Jacobson, I., & Booch, G. (2005). *The Unified Modeling Language Reference Manual* (2nd, Vol. 240). Boston, MA, Addison-Wesley.

Saaty, T. L. (1990). How to make a decision: The Analytic Hierarchy Process. *European Journal of Operational Research*, *48*, 577–591. https://doi.org/10.1007/978-1-4419-6281-2_31

Saleiro, P., Kuester, B., Hinkson, L., London, J., Stevens, A., Anisfeld, A., Rodolfa, K. T., & Ghani, R. (2018). Aequitas: A Bias and Fairness Audit Toolkit, (2018), arXiv 1811.05577. http://arxiv.org/abs/1811.05577

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J. F., & Dennison, D. (2015). Hidden technical debt in machine learning systems. *Advances in Neural Information Processing Systems*, *2015-Janua*, 2503–2511.

Selbst, A. D., Boyd, D., Friedler, S. A., Venkatasubramanian, S., & Vertesi, J. (2019). Fairness and abstraction in sociotechnical systems, In *Fat* 2019 - proceedings of the 2019 conference on fairness, accountability, and transparency*, New York, New York, USA, ACM Press. https://doi.org/10.1145/3287560.3287598

Sen, A., & Foster, J. (1997). *On Economic Inequality*. Oxford University Press.

Shapiro, D. L., Buttner, E. H., & Barry, B. (1994). Explanations: What factors enhance their perceived adequacy? https://doi.org/10.1006/obhd.1994.1041

Sheshasaayee, A. (2017). A Study of Automated Decision Making Systems. *International Journal of Engineering And Science*, *7*(1), 28–31. www.researchinventy.com

Simon, H. A. (1977). *The New Science of Management Decision* (3rd). Upper Saddle River, NJ, Prentice Hall PTR. https://doi.org/10.5555/540170

Simons, D. (2019). *Design for fairness in AI: Cooking a fair AI dish* (Doctoral dissertation). Delft University of Technology. Delft. https://repository.tudelft.nl/islandora/object/uuid:5a116c17-ce0a-4236-b283-da6b8545628c

Simons, L. P., & Verhagen, W. P. (2008). Applying value-sensitive design and quality function deployment to healthcare ICT: the case of Dutch primary care unit dossiers. *Journal of Design Research*, *7*(2), 155–176. https://doi.org/10.1504/JDR.2008.020853

Speicher, T., Heidari, H., Grgić-Hlača, N., Gummadi, K. P., Singla, A., Weller, A., & Zafar, M. B. (2018). A unified approach to quantifying algorithmic unfairness: Measuring individual & group unfairness via inequality indices, In *Proceedings of the acm sigkdd international conference on knowledge discovery and data mining*, New York, New York, USA, ACM Press. https://doi.org/10.1145/3219819.3220046

Srivastava, M., Heidari, H., & Krause, A. (2019). Mathematical Notions vs. Human Perception of Fairness: A Descriptive Approach to Fairness for Machine Learning, arXiv 1902.04783. http://arxiv.org/abs/1902.04783

Stoyanovich, J. (2019). TransFAT: Translating Fairness, Accountability and Transparency into Data Science Practice (D. Firmani, E. Nieddu, L. Tanca, & R. Torlone, Eds.). In D. Firmani, E. Nieddu, L. Tanca, & R. Torlone (Eds.), *International workshop on processing information ethically*, Rome.

Stoyanovich, J., & Howe, B. (2019). Nutritional Labels for Data and Models. *Data Engineering*, (1926250), 13.

Sun, S., Cao, Z., Zhu, H., & Zhao, J. (2019). A Survey of Optimization Methods From a Machine Learning Perspective. *IEEE Transactions on Cybernetics*, arXiv 1906.06821, 1–14. https://doi.org/10.1109/tcyb.2019.2950779

Sun, T., Gaut, A., Tang, S., Huang, Y., ElSherief, M., Zhao, J., Mirza, D., Belding, E., Chang, K.-W., & Wang, W. Y. (2019). Mitigating Gender Bias in Natural Language Processing: Literature Review.

arxiv.org, arXiv 1906.08976. https://arxiv.org/abs/1906.08976%20http://arxiv.org/abs/1906.08976

Sweeney, L. (2013). Discrimination in Online Ad Delivery. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.2208240

Tang, R., Du, M., Li, Y., Liu, Z., & Hu, X. (2020). Mitigating Gender Bias in Captioning Systems, arXiv 2006.08315v1. https://github.com/CaptionGenderBias2020.

Thew, S., & Sutcliffe, A. (2018). Value-based requirements engineering: method and experience. *Requirements Engineering*, *23*(4), 443–464. https://doi.org/10.1007/s00766-017-0273-y

Thibaut, J. W., & Walker, L. (1975). *Procedural justice: A psychological analysis* (1st). Hillsdale, L. Erlbaum Associates.

Thuan, N. H., Drechsler, A., & Antunes, P. (2019). Construction of design science research questions. *Communications of the Association for Information Systems*, *44*(1), 332–363. https://doi.org/10.17705/1CAIS.04420

Timan, T., & Grommé, F. (2020). A framework for social fairness ; Insights from two algorithmic decision-making controversies in The Netherlands.

Tyler, T. R., & Lind, E. . (1992). A relational model of authority in groups. *Advances in Experimental Social Psychology*, *25*(100), 115–191. https://doi.org/10.1016/S0065-2601(08)60283-X

van de Poel, I. (2013). Translating Values into Design Requirements. https://doi.org/10.1007/978-94-007-7762-0_20

van de Poel, I. (2015a). Conflicting Values in Design for Values (J. van den Hoven, P. E. Vermaas, & I. van de Poel, Eds.). In J. van den Hoven, P. E. Vermaas, & I. van de Poel (Eds.), *Handbook of ethics, values, and technological design: Sources, theory, values and application domains*. Dordrecht, Springer Netherlands. https://doi.org/10.1007/978-94-007-6970-0

van de Poel, I. (2015b). Design for Values in Engineering (J. van den Hoven, P. E. Vermaas, & I. van de Poel, Eds.). In J. van den Hoven, P. E. Vermaas, & I. van de Poel (Eds.), *Handbook of ethics, values, and technological design: Sources, theory, values and application domains*. Dordrecht, Springer Netherlands. https://doi.org/10.1007/978-94-007-6970-0

van den Hoven, J., Vermaas, P. E., & van de Poel, I. (2015). *Handbook of ethics, values, and technological design: Sources, theory, values and application domains* (J. van den Hoven, P. E. Vermaas, & I. van de Poel, Eds.). Dordrecht, Springer Netherlands. https://doi.org/10.1007/978-94-007-6970-0

van der Velden, M., & Mörtberg, C. (2015a). Participatory Design and Design for Values (J. van den Hoven, P. E. Vermaas, & I. van de Poel, Eds.). In J. van den Hoven, P. E. Vermaas, & I. van de Poel (Eds.), *Handbook of ethics, values, and technological design: Sources, theory, values and application domains*. Dordrecht, Springer Netherlands. https://doi.org/10.1007/978-94-007-6970-0

van der Velden, M., & Mörtberg, C. (2015b). Value Sensitive Design: Applications, Adaptations, and Critiques (J. van den Hoven, P. E. Vermaas, & I. van de Poel, Eds.). In J. van den Hoven, P. E. Vermaas, & I. van de Poel (Eds.), *Handbook of ethics, values, and technological design: Sources, theory, values and application domains*. Dordrecht, Springer Netherlands. https://doi.org/10.1007/978-94-007-6970-0

Venable, J., Pries-Heje, J., & Baskerville, R. (2012). A Comprehensive Framework for Evaluation in Design Science Research: Advances in Theory and Practice, In *Proceedings of the 7th international conference on design science research in information systems*. https://doi.org/10.1007/978-3-642-29863-9_31

Verma, S., & Rubin, J. (2018). Fairness definitions explained, In *Proceedings - international conference on software engineering*. https://doi.org/10.1145/3194770.3194776

Verschuren, P., & Hartog, R. (2005). Evaluation in design-oriented research. *Quality and Quantity*, *39*(6), 733–762. https://doi.org/10.1007/s11135-005-3150-6

Vogelsang, A., & Borg, M. (2019). Requirements Engineering for Machine Learning: Perspectives from Data Scientists, arXiv 1908.04674. http://arxiv.org/abs/1908.04674

Wand, Y., & Wang, R. Y. (1996). Anchoring Data Quality Dimensions in Ontological Foundations. *Communications of the ACM*, *39*(11), 86–95.

Wang, R., Harper, F. M., & Zhu, H. (2020). Factors Influencing Perceived Fairness in Algorithmic Decision-Making, 1–14. https://doi.org/10.1145/3313831.3376813

Wang, Z., Qinami, K., Karakozis, I. C., Genova, K., Nair, P., Hata, K., & Russakovsky, O. (2019). Towards Fairness in Visual Recognition: Effective Strategies for Bias Mitigation, arXiv 1911.11834, 8919–8928. http://arxiv.org/abs/1911.11834

Whittaker, M., Crawford, K., Dobbe, R., Fried, G., Kaziunas, E., Mathur, V., Myers West, S., Richardson, R., Schultz, J., & Schwartz, O. (2018). AI Now Report 2018. *AI Now*, (December), 1–62. https://ainowinstitute.org/AI%7B%5C_%7DNow%7B%5C_%7D2018%7B%5C_%7DReport.pdf

Williamson, R. C., & Menon, A. K. (2019). Fairness risk measures. *36th International Conference on Machine Learning, ICML 2019*, *2019-June*arXiv 1901.08665, 11763–11774.

Wood, M. R. (2013). Why youŕe confusing frameworks with methodologies. https:%5C/%5C/www.projectmanagement.com%5C/articles%5C/278600%5C/Why-Youre-Confusing-Frameworks-with-Methodologies

Yaghini, M., Heidari, H., & Krause, A. (2019). A Human-in-the-loop Framework to Construct Context-dependent Mathematical Formulations of Fairness, arXiv 1911.03020. http://arxiv.org/abs/1911.03020

Zafar, M. B. (2019). Discrimination in Algorithmic Decision Making : From Principles to Measures and Mechanisms A dissertation submitted towards the degree Doctor of Engineering of the Faculty of Mathematics and Computer Science of by
only looks at group fairness: disparate treatment, disparate impact - within-group fairness - procedural fairness - representative harms.

Zhu, H., Yu, B., Halfaker, A., & Terveen, L. (2018). Value-sensitive algorithm design: Method, case study, and lessons. *Proceedings of the ACM on Human-Computer Interaction*, *2*(CSCW), 1–23. https://doi.org/10.1145/3274463

Zliobaite, I. (2015). A survey on measuring indirect discrimination in machine learning, arXiv 1511.00148. http://arxiv.org/abs/1511.00148

Žliobaitė, I. (2017). Measuring discrimination in algorithmic decision making. *Data Mining and Knowledge Discovery*, *31*(4), 1060–1089. https://doi.org/10.1007/s10618-017-0506-1

Zowghi, D., & Coulin, C. (2005). Requirements Elicitation: A Survey of Techniques, Approaches, and Tools, In *Engineering and managing software requirements*.