Assuring Fairness of Algorithmic Decision Making

Marc P. Hauer
Algorithm Accountability Lab
TU Kaiserslautern, Germany
OrcID: 0000-0002-1598-1812
Email: hauer@cs.uni-kl.de

Rasmus Adler
Fraunhofer IESE
Kaiserslautern, Germany
OrcID: 0000-0002-7482-7102
Email: Rasmus.Adler@iese.fraunhofer.de

Katharina Zweig
Algorithm Accountability Lab
TU Kaiserslautern, Germany
OrcID: 0000-0002-4294-9017
Email: zweig@cs.uni-kl.de

***Abstract*—Assuring fairness of an algorithmic decision making (ADM) system is a challenging task involving different and possibly conflicting views on fairness as expressed by multiple fairness measures. We argue that a combination of the agile development framework *Acceptance Test-Driven Development* (ATDD) and the concept of *Assurance Cases* from safety engineering is a pragmatic way to assure fairness levels that are adequate for a predefined application. The approach supports examinations by regulating bodies or related auditing processes by providing a structured argument explaining the achieved level of fairness and its sufficiency for the application.**

I. Introduction

Algorithms are increasingly involved in systems used to support decision making. Prominent application examples for algorithmic decision making (ADM) [1] are supporting sentencing in the criminal justice system [2], product recommendation [3], medical diagnosis and treatment [4] or job applicant pre-selection [5]. It is an extra-functional requirement that ADM systems adhere to all legal standards regarding discrimination. While there is no clear, quantifiable model of how to measure the extent of discrimination from the legislator's perspective [6, p. 5], computer science has come up with about 20 fairness measures for this task [7].

In order to control for existing or upcoming discrimination, e.g. because of data drift, it is not enough to assess the fairness of a system once; instead, a continuous, digital assessment of the fairness of the decisions is mandatory. In other words, there needs to be a process that **assures the fairness of the decisions**.

In traditionally programmed algorithms, this requires as a first step that a group of people generates a comprehensible common understanding of the term "fairness" and its specification, which is then implemented. The disadvantage of this quantification process is that it is very hard to find a formula that complies with the group's intuition about fairness in all possible cases or to even consider all possible cases. In contrast to traditionally programmed algorithms, machine learning offers the advantage that 'fairness' does not need to be specified upfront, but can be demonstrated by a carefully chosen set of examples in a training data set. The flip side of the coin is that it is hard to assure whether the algorithm has actually learned what fairness is. Formal proofs are not applicable as a related formal specification is missing. Reviews can also not be applied as AI-based systems are hardly explainable or comprehensible. That means even if an AI system made decisions that are deemed fair so far, without being able to explain its behaviour, it cannot be assured that it does the right thing for the right reasons. Therefore, the AI system might make a wrong decision in future situations. In spite of these challenges for assuring fairness, ADM systems are increasingly used for making critical decisions, which leads to the open problem of how to assure fairness. This is different from assuring a discrimination-free ADM system, but rather follows the idea of assuring protection from unreasonable risk, which is common in safety engineering. An important basis for assuring and arguing sufficient fairness is testing. This holds true in particular if reviews and other traditional quality assurance methods cannot be used when machine learning is involved.

We thus propose to combine approaches from testing and safety engineering for assuring fairness of AI-based decisions: First, we suggest to use an *Acceptance Test-Driven Development* (**ATDD**) approach, which aims at the derivation of testable acceptance criteria for a given use case. Second, we propose to develop an *Assurance Case* explaining which level of fairness is achieved with the ATDD approach and why this level is sufficient for the use case. ISO/IEC/IEEE 15026-1:2019 defines an Assurance Case as a reasoned, auditable artefact, which allows to validate a claim—in this case, e.g., that the system achieves a sufficient level of fairness in the use case. It does so by structuring the underlying assumptions and the necessary evidence for it in such a way that the truthfulness of the claim can be logically induced from the provided evidence.

We start with a short introduction of Acceptance Test-Driven Development and Assurance Cases.

II. ATDD and Assurance Cases

This section gives a short introduction of the two main frameworks proposed to assure fairness of a particular ADM system in a particular use case, namely Acceptance Test-Driven Development (ATDD) and Assurance Cases.

A. ATDD

ATDD is a framework that aims for deriving concrete, unambiguous acceptance criteria by developing example situations that elaborate how a system should work. The examples are specified by a team of stakeholders (usually at least a business expert/customer representative/product owner, a developer and a tester [8]), which work together to establish a ubiquitous understanding of the wanted behavior of a system. In the context of ATDD, various formats are proposed for stakeholder meetings to jointly develop requirements, such as a specification workshop [9], [10].

For larger systems, it might be sufficient to start producing code for the first ATDD requirements, while producing further examples in parallel. If a software product is supposed to change over time, the examples can be extended or adjusted to match up-to-date requirements. Eventually, multiple iterations of meeting and building up examples might be necessary to find an encompassing set of all necessary cases. This way, the concept fits into common agile practices.

The collected example situations are then used to define the acceptable behavior of the system in each of these situations and to abstract from them so-called *acceptance criteria*, which need to be observable.

Finally, these acceptance criteria are formulated as tests, while prioritizing those that can be automated.

This basic framework of ATDD can be found in numerous variations with different nuances or adaptations of techniques and under multiple names, like Behavior-Driven Development (BDD) [11] or Specification by Example [10]. An extensive explanation of the various facets of the concept can be found in [12].

B. Assurance Cases

An Assurance Case is a reasoned and compelling argument, supported by a body of evidence, which states that a system, service or organisation will operate as intended for a defined application in a defined environment. They are heavily and increasingly used for assuring safety of ADM systems like autonomous vehicles. For instance, the safety standard UL 4600 refers to the development of an Assurance Case. A claim for an AI-based perception system might be: "The systems detects pedestrians sufficiently reliable within its operational design domain (ODD)". The ODD specifies the operating conditions under which a system or feature is designed to function; for instance, geographical time-of-day, or weather conditions. The claim itself is based on *evidence*. These include results concerning simulation, evaluation of similarity between reality and specification of validation data, evaluation of run-time monitoring, and so on. The main task of the Assurance Case is to justify why and under which assumptions the evidence imply the claim. To this end, the main claim is decomposed into sub-claims that are either also based on the fulfillment of hierarchically structured sub-claims or that can be directly induced from evidence. As illustrated in Fig. 1a, each decomposition of a claim is made explicit by an *argument* that explains the idea behind a decomposition. Furthermore, all relevant *assumptions* for concluding that the sub-claims imply the claims are made explicitly and connected to the argument. To ease the understanding of an argument, contextual information can be attached to it as well. There are several notations available for modeling an Assurance Case and the modeling features are sightly different. In this paper, we do not stick to a specific notation and focus on the concept of structured argumentation.

The hierarchically structured argumentation helps to:

1) manage complexity of the argumentation
2) foster the generation of meaningful evidence with a top-down decomposition
3) avoid overstatements about the achieved level of fairness (complementary bottom-up development)
4) to ensure completeness and correctness by enabling a rigorous review or audit of each argument

In summary, an Assurance Case provides an argumentation framework where the statement that the evidence supports the claim under the given assumptions can be reasoned and understood. In the next section, we will now integrate both concepts to sketch how fairness of an ADM-system can be assured.

III. Fusing ATDD with assurance cases

We propose to combine Acceptance Test-Driven Development (ATDD) with Assurance Case development to create a process suitable for identifying and testing complex extra-functional requirements on the example of fairness. While ATDD aims for finding acceptance criteria based on specific examples elaborated by a diverse group of stakeholders and to test for them, an Assurance Case argues that the criteria found with the help of ATDD are adequate and makes the otherwise implicit assumptions visible. It provides a framework for structured argumentation and it documents the results of discussions, whereby it reveals misconceptions and shortcomings.

Fairness as an extra-functional requirement first necessitates a decision of what actually is the system that makes decisions. We have argued elsewhere that it is not enough to focus on the ADM system to understand whether fair decisions are being made but that the final, social process in which the decision making is embedded needs to be checked for fairness [13]. In the following, we will first describe the necessary adaptation of ATDD to fairness and then go on to the specifics of how fairness requirements could be stated as an Assurance Case.

A. Adaptations of the ATDD process

We argued in the introduction that there is no way to find a complete, consistent, quantified specification of fairness. However, in a given situation, it is possible to evaluate the list of all known fairness measures, select a subset of them and discuss suitable thresholds to fulfill a specific goal. For example, a company might want to use an ADM system during a hiring process while reducing the risks for lawsuits. Thus, the goal is not to make the ADM system free of discrimination (which is impossible in principle) but only to make the development and usage of the system robust enough to reduce the risk to an acceptable level. Thus, during the ATDD process, the expert groups need to find situations that might be perceived as being unfair or discriminatory and select the fairness measures to identify these situations by setting a threshold after which the decision making is ruled as problematic.

Phase 1: Composition of the Expert groups: To make this procedure robust, a larger set of experts needs to be considered: In addition to the traditional stakeholders recommended by the literature for requirements engineering, we propose to consider legal experts, data scientists and ethicists as well to make sure that all extra-functional requirements regarding fairness have been identified. Moreover, in the context of discrimination, it is especially important that members of all respective groups are invited to collect example situations from as many perspectives as possible.

In such an interdisciplinary setting, it is essential that any current or future stakeholder can quickly understand the intent of each example: There are multiple techniques how to consistently write down examples. The best fitting method needs to be derived by experience and to depend on the specific project, though there are some techniques especially popular, like the Gherkin Syntax [14], tabulated formats [14] or keywords [15].

Phase 2&3: Identification of Acceptance Criteria and Definition of Tests based on Fairness Measures: Once the example situations are collected, they need to be grouped in a way in which they can be identified using a fairness measure. If there is a situation for which there is no fairness measure or other means to identify it, this needs to be documented. In general, the choice of concrete fairness measures is not trivial and must be discussed on a case-by-case basis. Each choice brings implications that the participants must be aware of in order to make informed decisions which include all perspectives from societal requirements to ethical judgement and lawfulness.

With the fairness measures selected, acceptance criteria need to be derived. The choice of the respective acceptance criteria needs to be made such that they are satisfiable: Often, different fairness measures are in conflict with each other, i.e., both cannot be optimized at the same time [16]. Similarly, fairness and quality of decisions cannot always be optimized simultaneously.

In any case, regarding fairness, it is always necessary to adapt an acceptance criterion to the various ways in which the data is sampled from the population, among others for the following reasons:

1) In many cases, sensitive attributes create groups with very unequal sizes. If the population is sampled with respect to the ratios in which they belong to the different groups and the fairness measure is averaged over all, the wrong decisions about minorities count less than those of the majority group. However, if the test data set is balanced, i.e., all groups are tested by the same number of decisions, the fairness measure is more balanced. However, in reality, more people of the majority group are decided upon and thus, the fairness measure does not meet reality. Therefore, there might be two thresholds, one for the realistic distribution, and one for the balanced sample distribution in order to assure fairness of the system.

2) Another aspect of input data is that there might be correlations between sensitive data and data that is relevant for the decision. In the job application data, gender might be statistically related to success at school. In such a case, a test searching for unfair treatment based on school grade may be reasonable. Different thresholds for the whole data set and the various conditions might be wanted in such a case.

Next to giving thresholds for single fairness measures, another possibility is to create formulas that build a weighted average over multiple fairness-measure values.

The test situations could also be based on input data sets which are designed in a counter-factual approach [17]: There are different concepts under the same name, but all have in common that the system's function under a designed data set is defined. For example, in a job application system, it might be required that the result of the decision making is not changed by, e.g., the gender of a person. This can be tested by exchanging the gender in real job applications and testing the outcome with various fairness measures.

There might be even more ways in which fairness measures can be used to assure fairness of a system: for the ATDD and Assurance Case approach, it is only necessary that they can be designed as tests.

The consensus achieved in the meetings should be documented. This includes in particular the arguments explaining why the ADM is fair enough if it fulfills the acceptance criteria. For this purpose and for demonstrating that the acceptance criteria are actually fulfilled, we propose Assurance Cases, a best practice which has already proven its value in the field of safety.

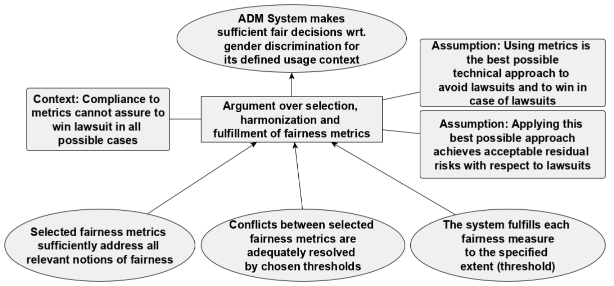
The acceptance criteria can then be used as a basis for developing tests that become part of the Assurance Case.

B. Development of Assurance Cases for fairness requirements

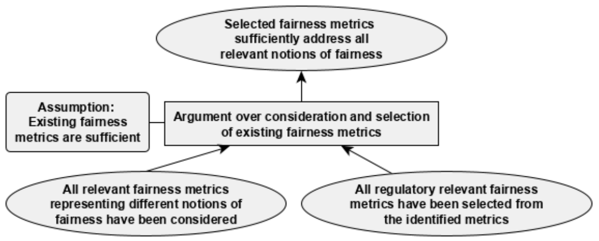
The Assurance Case is more than just a combination of different fairness-related tests. It starts with the main-claim, namely the statement that the system is "fair enough". This main-claim depends on the situation it is meant for. Thus, the Assurance Case adds an argument to the main-claim that describes the situation (context information) and that describes the conditions under which it can be structured into sub-claims. In Fig. 1a, as a running example, the situation is a job-application system and the goal is for it to be fair enough to minimize the risk of lawsuits. The context providing necessary background information and the assumptions under which the argument is valid are visualized as side nodes.

The sub-claims, on the one hand, contain the tests developed in the ATDD approach. However, it also needs to be assured (at evaluation time) that the conditions, under which the tests were deemed to be correct, are still valid. For example, a sub-claim to assure fairness of the entire system would need to be that no new fairness measures to be considered have been published. Secondly, that all known conflicts between fairness measures are still the same.

Assurance Cases as means of external validation: To assure that a system has a certain property, the way this is evaluated (for example, by an Assurance Case) also needs to be part of the assessment. In our case, for example, the selection process of fairness measures to be considered, warrants some


ADM System makes sufficient fair decisions wrt. gender discrimination for its defined usage context
Context: Compliance to metrics cannot assure to win lawsuit in all possible cases
Argument over selection, harmonization and fulfillment of fairness metrics
Assumption: Using metrics is the best possible technical approach to avoid lawsuits and to win in case of lawsuits
Assumption: Applying this best possible approach achieves acceptable residual risks with respect to lawsuits
Selected fairness metrics sufficiently address all relevant notions of fairness
Conflicts between selected fairness metrics are adequately resolved by chosen thresholds
The system fulfills each fairness measure to the specified extent (threshold)

(a) The main claim of the Assurance Case is divided into multiple sub-claims


Selected fairness metrics sufficiently address all relevant notions of fairness
Assumption: Existing fairness metrics are sufficient
Argument over consideration and selection of existing fairness metrics
All relevant fairness metrics representing different notions of fairness have been considered
All regulatory relevant fairness metrics have been selected from the identified metrics

(b) Assurance Case path arguing over the selection of fairness measures

Fig. 1: Example excerpts from an assurance case graph

quality as well. The conditions and assumptions under which this selection took place (as part of phase 1 in the ATDD), can also be described in the Assurance Case. For example, the first two sub-claims in Fig. 1a make the selection process of the considered fairness measures transparent and allow assurance of its correct conductance. Fig. 1b shows a more detailed picture of how the sub-claim that the selection process was adhered to could be further divided into other sub-claims and finally evidence to assure it.

IV. Conclusion and Future Work

The proposed approach is not a deterministic process and will thus not lead to a unique result. It can also not solve the main and principle problem of how to define fairness in a quantified and widely accepted manner. Nevertheless, it describes a pragmatic approach to come to a well-documented argument about when and under which assumptions a system is “fair enough” to be used.

By modelling the argumentation about why the solutions and the fulfillment of the chosen tests confirm the achievement of the objectives as an Assurance Case, the rationales for decisions can be documented and disclosed for an external review or audit, for example, in a lawsuit.

By employing the approach before development, and by automating the tests and documenting the results, this process yields the potential to provide a long-term protection against unwanted changes, e.g. through further training or errors when changing the code.

Although the article explicitly focuses on fairness, the process described can be applied to the testing of extra-functional requirements like ethics in general. This makes sense wherever clear legal or normative requirements still seem to be a long way off. Whether the process is suitable for ethical testing must be determined in industrial trials.

Acknowledgment

This work was supported by the German Federal Ministry of Labour and Social Affairs (BMAS) in the project **ExamAI—KI Testing and Auditing** (Funding code DKI.00.0002_3.20). We thank our research partners from the Gesellschaft für Informatik, the Stiftung Neue Verantwortung, and Prof. Dr. Borges and his team from the University of Saarbrücken for their helpful comments on an earlier version of this work.

References

[1] N. Diakopoulos, “Accountability in algorithmic decision making,” *Communications of the ACM*, vol. 59, no. 2, pp. 56–62, 2016.

[2] J. Angwin, J. Larson, S. Mattu, and L. Kirchner, “Machine bias,” *Propublica, see https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing,*, 2016.

[3] D. Mattioli, “On Orbitz, Mac users steered to pricier hotels,” *Wall Street Journal*, vol. 23, p. 2012, 2012.

[4] W. R. Robinson, A. Renson, and A. I. Naimi, “Teaching yourself about structural racism will improve your machine learning,” *Biostatistics*, vol. 21, no. 2, pp. 339–344, 2020.

[5] J. Dastin, “Amazon scraps secret AI recruiting tool that showed bias against women,” *Reuters*, 2018.

[6] S. Wachter, B. Mittelstadt, and C. Russell, “Why fairness cannot be automated: Bridging the gap between eu non-discrimination law and AI,” *Available at SSRN*, 2020.

[7] S. Verma and J. Rubin, “Fairness definitions explained,” in *2018 IEEE/ACM International Workshop on Software Fairness (FairWare)*. IEEE, 2018, pp. 1–7.

[8] A. Cockburn, *Agile software development: The cooperative game*. Pearson Education, 2006.

[9] G. Adzic, *Bridging the communication gap: specification by example and agile acceptance testing*. Neuri Limited, 2009.

[10] ——, “Specification by example,” *Book your training with Díaz & Hilterscheid!*, p. 20, 2011.

[11] C. Solis and X. Wang, “A study of the characteristics of behaviour driven development,” in *2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications*. IEEE, 2011, pp. 383–387.

[12] M. Gärtner, *ATDD by example: a practical guide to acceptance test-driven development*. Addison-Wesley, 2012.

[13] K. A. Zweig, G. Wenzelburger, and T. D. Krafft, “On chances and risks of security related algorithmic decision making systems,” *European Journal for Security Research*, vol. 3, no. 2, pp. 181–203, 2018.

[14] E. C. dos Santos and P. Vilain, “Automated acceptance tests as software requirements: An experiment to compare the applicability of fit tables and gherkin language,” in *International Conference on Agile Software Development*. Springer, 2018, pp. 104–119.

[15] R. Hametner, D. Winkler, and A. Zoitl, “Agile testing concepts based on keyword-driven testing for industrial automation systems,” in *IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society*. IEEE, 2012, pp. 3727–3732.

[16] G. Pleiss, M. Raghavan, F. Wu, J. Kleinberg, and K. Q. Weinberger, “On fairness and calibration,” *Advances in Neural Information Processing Systems*, vol. 30, pp. 5680–5689, 2017.

[17] J. M. Nicklin, R. Greenbaum, L. A. McNall, R. Folger, and K. J. Williams, “The importance of contextual variables when judging fairness: An examination of counterfactual thoughts and fairness theory,” *Organizational Behavior and Human Decision Processes*, vol. 114, no. 2, pp. 127–141, 2011.