Bias-aware guidelines and fairness-preserving Taxonomy in software engineering education

Paola Spoletini
SWEGD, Kennesaw State University
Marietta, GA 30060, USA
pspoleti@kennesaw.edu

Reza M. Parizi
SWEGD, Kennesaw State University
Marietta, GA 30060, USA
rpariz1@kennesaw.edu

***Abstract*—This innovative practice work in progress paper tackles the problem of unfairness and bias in software, that recently has emerged in countless cases. This unfairness can be present in the way software makes its decision or can limit the software functionalities to work only with certain populations. Well-known examples of this problem are the Microsoft Kinect facial recognition algorithm, which does not work properly with darker skin players, and the software used in 2016 by Amazon.com to determine the parts of the United States to which offer free same-day delivery that made decisions that prevented minority neighborhoods from participating in the program. The reasons behind these phenomena have often roots in the fact that software is created by humans who are biased and live in biased and non-inclusive environments. Recent research from the software engineering community is starting to tackle this problem at many levels from requirements analysis to the new automatic fairness testing technique (proposed first at FSE 2017 conference). However, research in bias of software is still a very undervalued and rarely discussed problem as software is often seen as a product immune to bias and non-inclusivity. This problem will be not addressed unless software engineering educators start to include this notion as a first-class problem in their foundation courses to future generation of scholars. In this work, we propose a set of bias-aware guidelines and taxonomy on how to flesh out this problem and possible solutions to it in software engineering curricula.**

***Index Terms*—Software engineering education, Fairness, Software engineering curricula**

I. Introduction and motivation

In our society software is everywhere: cars, houses, cellphones, and many other non-traditional devices. The amount of available software services and applications in the form of web services and mobile apps has been dramatically increasing over the years. Moreover, the advances in artificial intelligence and machine learning together with the large quantity of available data are leading to a shift in how software is used, enabling the software to make autonomous decisions. This causes in many cases the substitution of humans with software. There are already many examples of such software from recommender systems to decision makers. Software can now decide who gets a loan [18], what we "want" to buy [15], medical diagnoses and treatments [21], and can also drive cars, possibly damaging properties or injuring humans [7], [10]. Software has also a predominant role in every stage of the criminal justice systems up to the point to determine who goes to jail and who is set free [2], [8]. So, what if software is not fair and discriminate (as human beings do)?

In the general perception, software does not discriminate, because differently from human beings, software does not have biases and does not relies on stereotypes to make its decisions. However, that it is (not surprisingly) not true. As software is developed by human beings, even though through a rigorous process and relying on many tools, the biases that the designers and developers have can be transferred consciously or unconsciously to the software they produce. A well-known example of this problem is the software used in 2016 by Amazon.com to determine the parts of the United States to which offer free same-day delivery that made decisions that prevented minority neighborhoods from participating in the program [13], [15]. Moreover, the lack of diversity and the failure in considering all the relevant stakeholders might result in software that discriminates and does not work properly with certain populations. An example of this problem is the Microsoft Kinect facial recognition algorithm, which did not work properly with darker skin players.

During the Spring Semester 2018, we briefly introduced fairness testing [9] to the graduate students in the Software Testing course at Kennesaw State University (KSU), who were very surprised to hear about this problem. In the meantime, this problem has also been shared with other campus units (not in the computing field) and they had even a more surprised reaction. This was a wakeup call for us: raising awareness about the impact of bias, discrimination and lack of diversity in software and providing to our students the tools to prevent it and to identify unfairness when present are fundamental goals for our society in which software is ubiquitous. These goals are also in line with both the concerns raised in 2016 by the U.S. Executive Office of the President [6], which identified bias in software as a major concern for civil rights and one of the ten principles that Satya Nadella has laid out for artificial intelligence "AI must guard against bias, ensuring proper, and representative research so that the wrong heuristics cannot be used to discriminate" [17], unfortunately not currently implemented (e.g., [5]).

In this work, we are not proposing any new technique to check for or developing with fairness, but the design of modules to introduce in the main courses in undergraduate software engineering curricula definitions, techniques, analyses, and

978-1-5386-1174-6/18/$31.00 ©2018 IEEE

reflections on the problem of unfairness and discrimination in software, with the goal of raising the awareness and increasing the abilities of new software engineers. The reminder of the paper will present the analysis of current curricula in software engineering with respect to ethics and software fairness (Section II) and the description of the modules we plan to include in our curriculum (Section III). To conclude the paper we will reflect on the impact that this work could have on other curricula and other possible future work (Section IV).

II. (Preliminary) Analysis of Software Engineering Undergraduate Curricula

To evaluate the role of ethics, and in particular of fair software, we have analyzed 20 curricula of Bachelor of Science in Software Engineering (BS SWE) offered in United States Institutions [4]. These BS SWE are all ABET accredited [1] and, for their accreditation, they clearly outlined the Program Educational Objectives (PEOs), which represent the competencies and abilities that a graduate will have developed after around 5 years from graduation, and the Student Outcomes (SOs), which are the competencies and abilities upon graduation. Analyzing the PEOs and the SOs of the considered institutions, there are recurrent references to ethics. Notice that ABET requires accredited programs to include the following SO: (Students will have) "an ability to design a system, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, *ethical, health and safety*, manufacturability, and sustainability." (which is included as it is in the SOs of CalPoly, SJSU, MSOE, Penn State) This same SO can assume different flavors or be complemented with other related SOs, such as

- (Students will be able to) "Recognize the *social, professional, cultural, and ethical issues* involved in the use of computer technology and give them due consideration in decision making" (University of Miami, OH);
- (Students will have) "An understanding of professional and *ethical responsibility*" (e.g., CalPoly, SJSU);
- (Students will have) "An understanding of the critical role played by software systems, the professional responsibilities of software engineers, and *ethical issues that may be encountered in engineering practice*" (MSOE);
- (Students will have) "An understanding of professional, ethical, legal, security, and societal issues and responsibilities appropriate to the discipline" (Auburn);
- (Students will be able to) "Design appropriate solutions in one or more application domains *using software engineering approaches that integrate ethical, social, legal, and economic concerns*" (UCI);
- (Students will have) "An understanding of and a commitment to address professional and *ethical responsibilities including a respect for diversity*" (Oregon State).

The SOs are a mean to satisfies the PEOs which have to be in line with the mission of the Institution. In most of the analyzed institutions, also the PEOs contain an explicit reference to ethics:

- (Graduates will be able to) "Demonstrate an understanding of the context and broader impacts of technology in their organization by, for example, engaging stakeholders outside their immediate team, or *by identifying ethical, economic, cultural, legal or environmental issues* related to work projects." (University of Miami, OH);
- (Graduates will be able to) "encourage and support *diversity and inclusiveness* in their workplace" (Iowa State);
- (Graduates are expected, within a few years of graduation, to have) "demonstrated an ability to be effective as both an individual contributor and a member of a development team with professional, *ethical, and social responsibilities*" (RIT);
- "Graduatess will be able to *identify ethical issues* related to the development of computer software (Michigan).

The BS SWE at KSU includes the SO mandated by ABET and also includes the following PEO: "Graduates will utilize and exhibit strong communication and interpersonal skill as well as professional and *ethics principles* when functioning as members and leaders of multi-disciplinary teams". In order to satisfy this outcome, our students have to take a course on Ethics and Professional Practices, that covers historical, social and economic consideration of the discipline, studies of professional conduct, risks, and liabilities, and intellectual property relative to the software engineering and computing professions. Moreover, in the program, (1) the role of bias in eliciting requirements is discussed in "Software System Requirements", (2) the importance of inclusion and diversity in development teams is discussed in many courses, and (3) accessibility and usability are studied in "User Centered Design". However, at the moment, we do not explicitly show to the students the result of bias and discrimination on software and we do not systematically tackle this problem in the different stages of software development lifecycle.

Most of the programs we have analyzed have a similar approach to include ethics-related issues into their programs, and none of the considered courses explicitly analyze the problem of bias and discrimination, and its effect on software products. The objective of these courses is usually to reflect on critical domains by considering safety and security issues, professional ethics and behavior, code of ethics, being a member of a team and participating in a professional manner, and professional responsibility to be able provide fair and honest peer evaluations.

III. Teaching How to Develop Fair Software

The Bloom's taxonomy of education objectives is a framework for classifying statements of what we expect students to learn as a result of instruction. The original taxonomy [3] provided definitions for each of the six major categories in the cognitive domain: knowledge, comprehension, application, analysis, synthesis, and evaluation. The categories were ordered from simple to complex and from concrete to abstract, and it was assumed that the taxonomy represented a cumulative hierarchy (i.e., mastering simpler categories is needed to be able to master more complex categories). The original

TABLE I
FAIRNESS IN THE BS SWE AT KSU THROUGH THE KNOWLEDGE CATEGORIES IN THE REVISED BLOOM'S TAXONOMY

Table Bloom's Taxonomy Levels	**Courses**					
	Intro to Software Engineering	***Software System Requirements***	***Software Design***	***Software Testing***	***User Centered Design***	***Capstone***
Factual Knowledge	Introduction to the problem of fairness and discrimination in software engineering; Examples from the literature (e.g., [8], [13], [15]).	Introduction to bias in the RE phases (from elicitation to the requirements specification document); Bias and user stories; Inclusion.	Discussion on how design can be affected by bias and lack of diversity; The role of external "components"	Introduction to fairness in testing [9].	Introduction to the central role of the user, the effect of stereotyping, and lack of diversity.	Recap of the impact of bias, discrimination, and lack of diversity in the software development process.
Conceptual Knowledge	Analysis of the types of biases and problems; Overview of the methods to address the problem; How bias, discrimination and lack of diversity can affect the software development lifecycle (e.g., [19], [20]).	Analysis of the issues related to bias and discrimination in the different phases of the requirements engineering process (e.g., [11], [24]).	Analysis of the consequences of developing with bias and its impact on composition and reuse.	Analysis of the formal definitions related to software fairness and their application on practical examples.	Analysis of different users and and discussion on their different needs, accessibility, inclusions and usability.	Analysis of possible biases in the assigned/chosen project.
Procedural Knowledge	N/A	Practice with different elicitation methods (e.g., interviews and questionnaires) to learn how to identify and avoid bias. Use of goal modeling analysis (e.g., [12]) to check requirements against bias and discrimination.	From the requirements to the development: (manual) analysis of the developed models.	Practice with fairness testing: algorithms and applications [9].	Use of inclusive usability testing as a tool to prevent inaccessibility and discrimination (e.g., [16]); Practice with heuristic evaluation: application to inaccessible systems.	Application of all the learnt techniques (when appropriate) in the capstone project.
Metacognitive Knowledge	N/A	Reflection on the role playing exercises and and the lessons learnt useful in the professional practice.	Reflection on biased and discriminating models; Brainstorming on how to deal with these models in the work environment and possible steps forward.	Reflection on the testing result on real(istic) examples and on when to apply it in real-life.	Analysis of an inaccessible system and reflection on the consequence for the discriminated users.	Reflection on the impact of the fairness-related technique in the development of a project and in the profession of software engineer.

taxonomy is mono-dimensional and mixed up noun (e.g., knowledge) and verbs (e.g., analyze). These problems have been solved in the revisited taxonomy [14], where knowledge is considered as an additional dimension and the thinking skills are characterized only by verbs. In the revised taxonomy, knowledge has been categorized in: (1) Factual Knowledge, that comprises the basic elements that students must know to be acquainted with a discipline; (2) Conceptual Knowledge, i.e., the knowledge of the relationships among the basic elements within a larger structure that enable them to function together; (3) Procedural Knowledge, i.e., the knowledge of how to do something; and (4) Metacognitive Knowledge, i.e., the knowledge of cognition as well as awareness and knowledge of ones own cognition. The thinking skills are remember, understand, apply, analyze, evaluate, create.

The first step to include the concept of unfair and discriminating software in the BS SWE at KSU is to include a discussion on it in the Ethics and Professional Practices course. However, this course follows the discussion-based approach, while one of our goals is also to equip students with the necessary tools to avoid the development of unfair and discriminating software and the ability to spot such a software, if present. For this reason, we also have identified 6 courses in which we plan to introduce fairness-related concepts and techniques to prevent unfairness and discrimination in the software development lifecycle, and ultimately in software product, and identify them, when present. In particular, we plan to introduce the new content in Introduction to Software

Engineering, Software Systems Requirements, Software Design, Software Testing, User Centered Design, and Capstone. We have identified the notions and possible tools to introduce in these courses mapping them on the different categories of knowledge, as reported in Table I. These activities have the goal of fulfilling the following student outcomes, that will contribute to satisfy our ethics-related SO and PEO in a broader way.

We are expecting that, after being exposed to all the content and activities presented in Table I, the students will be able to

- Define the possible types of problems created by bias, discrimination, and lack of diversity in the software development lifecycle and on software products;
- Describe how unfairness can be addressed in the software development lifecycle;
- Identify the appropriate approach(es) to adopt to prevent unfairness and discrimination in software depending on the domain and the settings;
- Use ad-hoc and general requirements, design, and testing techniques to develop fair software;
- Analyze and test existing software to spot unfairness and discrimination;
- Create awareness on the problem of software unfairness.

A. (Research) Agenda

In order to evaluate the choices for our curriculum, we will integrate this content starting with "Introduction of Software Engineering" and "Software Systems Requirements" in Fall 2018. We will keep implementing these changes (two new courses a semester) and we plan to have all the content in place by Summer 2019. By that time, we will have a cycle of students who went through all the courses. In order to evaluate the impact of the added material, we will set up a set of questionnaires to measure the changes in awareness and capability of intentionally intervene to prevent unfair software products. We will also gather the interest of the students in the process and their feedback to improve the modules through focus groups. The new content will also be evaluated using traditional assessment tools such as homework and specific exercises in the tests. After three iterations of the process on campus (estimated date: Spring 2020), we will develop online lectures and material that can be disseminated and incorporated also in similar courses in other institutions.

IV. Conclusion and Future Work

In this paper we have described our plan to include the problem of fairness in software in our BS SWE curriculum and an agenda to implement and evaluate it. The problem of unfairness and discrimination in software should be discuss also outside software engineering programs. The changes in our curriculum will impact also on the education of Computer Science students (who take both introduction to Software Engineering and Ethics and Professional Practices) and IT students (who take Ethics and Professional Practices). However, the problem could also be addressed in algorithms courses, by presenting algorithms such as [23]. Furthermore, we believe that this topic should be disseminated at campus-level; so, we plan to create one-hour seminars for the non-computing campus population.

References

[1] ABET, http://www.abet.org/

[2] Angwin J., Larson J., Mattu S., and Kirchner L., Machine bias. ProPublica, May 23, 2016. https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing

[3] Bloom, Engelhart, Furst, Hill, and Krathwohl. Taxonomy of Educational Objectives: The Classification of Educational Goals. Handbook I: Cognitive Domain, 1956.

[4] College choice. https://www.collegechoice.net/rankings/best-bachelors-in-software-engineering-degree/

[5] Crawford K., Artificial Intelligences White Guy Problem. The New York Times, June 25, 2016.

[6] Executive Office of the President. Big data: A report on algorithmic systems, opportunity, and civil rights. https://www.whitehouse.gov/sites/default/files/microsites/ostp/2016_0504_data_discrimination.pdf, 2016.

[7] Fagnant D.J. and Kockelman K., Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations. Transportation Research Part A: Policy and Practice, 2015, 77, pp. 167-181.

[8] Ferral K., Wisconsin supreme court allows state to continue using computer program to assist in sentencing. The Capital Times, July 13, 2016. http://host.madison.com/ct/news/local/govt-and-politics/wisconsin-supreme-court-allows-state-to-continue-using-computer-program/article_7eb67874-bf40-59e3-b62a-923d1626fa0f.html

[9] Galhotra S. , Brun Y. , and Meliou A., Fairness testing: testing software for discrimination. ESEC/FSE 2017, ACM, pp 498-510.

[10] Goodall N.J., Can you program ethics into a self-driving car? IEEE Spectrum, 2016, 53(6), pp.2858.

[11] Hofmann, H.F. and Lehner, F., Requirements engineering as a success factor in software projects. IEEE Software, 2001, 18(4), pp. 58-66.

[12] Jureta, I.J., Faulkner, S., and Schobbens, PY., Clear justification of modeling decisions for goal-oriented requirements engineering. Requirements Engineering Journal, 2008 13: 87.

[13] Ingold D. and Soper S.. Amazon doesnt consider the race of its customers. Should it? Bloomberg, April 21, 2016. http://www.bloomberg.com/ graphics/2016-amazon-same-day

[14] Krathwohl D.R., A Revision of Bloom's Taxonomy: An Overview, Theory Into Practice, 2010, 41(4), pp. 212-218

[15] Mattioli D., On Orbitz, Mac users steered to pricier hotels. The Wall Street Journal, August 23, 2012. http://www.wsj.com/articles/SB10001424052702304458604577488822667325882

[16] Mayhew D.J., The usability engineering lifecycle. CHI '99, Extended Abstract, ACM, pp. 147-148.

[17] Nadella S., The partnership of the future. Slate, June 28, 2016. http://www.slate.com/articles/technology/future_tense/2016/06/microsoft_ceo_satya_nadella_humans_and_a_i_can_work_together_to_solve_society.html

[18] Olson P., The algorithm that beats your bank manager. CNN Money, March 15, 2011. http://www.forbes.com/sites/parmyolson/2011/03/15/the-algorithm-that-beats-your-bank-manager/#cd84e4f77ca8

[19] Parasuraman R. and Manzey D.H., Complacency and Bias in Human Use of Automation: An Attentional Integration. Human Factors, 2010, 52(3), pp. 381-410.

[20] Skitka L.J., Mosier K.L., Burdick M., Does automation bias decision-making?, International Journal of Human-Computer Studies, 1999, 51(5), pp. 991-1

[21] Strickland E., Doc bot preps for the O.R. IEEE Spectrum, 2016, 53(6), pp. 3260.

[22] Sutcliffe A. G. , Maiden N. A. M. , Minocha S., and Manuel D. , Supporting scenario-based requirements engineering. IEEE Transactions on Software Engineering, 1998, 24(12), pp. 1072-1088.

[23] Tramer F., Atlidakis V., Geambasu R.,l Hsu D., Hubaux J-P, Humbert M., Juels A., and Lin H., FairTest: Discovering unwarranted associations in data-driven applications. IEEE European Symposium on Security and Privacy, 2017.

[24] Zave P. and Jackson M., Four dark corners of requirements engineering. ACM Transactions on Software Engineering Methodology. 1997, 6(1), pp. 1-30.