Editor: **Tim Menzies**
North Carolina State University
tim@menzies.us

The Engineering Mindset Is an Ethical Mindset (We Just Don't Teach It That Way… Yet)

Tim Menzies, Brittany Johnson-Matthews, David L. Roberts, and Lauren Alvarez

From the Editor

For "SE for AI" column applications, do you have a surprising result or industrial experience? Something that challenges decades of conventional thinking in software engineering? If so, e-mail a one-paragraph synopsis to tim@menzies.us (subject line: "SE for AI: Idea: [Your Idea]"). If that looks interesting, I'll ask you to submit a 1,000–3,000-word article (where each graph, table, or figure is worth 250 words) for review for IEEE Software. Note: Heresies are more than welcome (if supported by well-reasoned industrial experiences, case studies, or other empirical results). —*Tim Menzies*

THERE ARE FAR too many examples where software engineers have deployed artificial engineering (AI) models with dubious, even dangerous, ethical properties (see "On the Need for Ethical Software"). **<AU: Per magazine style, Table 1 was changed to a sidebar.>** How can we fix that?

Why Teach More Ethics?

A recent Stack Overflow survey of more than 100,000 developers[1] showed that barely half responded that they would decline to develop unethical software. In that same survey, a third of respondents said, "it depends." These signals point to a need for frameworks and tools to support developers in making ethical choices and training them to always apply those frameworks. As educators, we ask the following question: "How can we, while training future software engineers, also be training ethical decision makers?"

There are many excellent computer science (CS) **<AU: Kindly check that the expansion of CS is correct.>** subjects devoted to ethics that allow free access to all their materials. For example, in our graduate software engineering (SE) class, we get students to apply the "seven steps to ethical decision making"[2] to the numerous ethical case studies documented at onlineethics.org[3] (for more examples of this kind of excellent material, see Hill[4] and see "Other Work on Integrating Ethics Across the Computer Science Curriculum"). **<AU: Per magazine style, Table 2 was changed to a sidebar.>**

While we applaud the authors of that material, we are worried that ethics is often taught separate from

Digital Object Identifier 10.1109/MS.2022.3227597
Date of current version: xxxxxx

 0740-7459/23©2023IEEE

ON THE NEED FOR ETHICAL SOFTWARE

Chapter Six of Safiya Noble's book *Algorithms of Oppression*[S1] **<AU: Please note that references cited only in the sidebars were renumbered as [S1], [S2], and so forth. Please check throughout.>** tells the sad tale of how a design quirk of Yelp ruined a small business. As one of Noble's interviewees put it, "Black people don't 'check in' and let people know where they're at when they sit in my (hair dressing salon). They already feel like they are being hunted; they aren't going to tell the Man where they are." Hence, that salon fell in the Yelp ratings (losing customers) since its patrons rarely pressed the "checked-in" button. There are many other examples where software engineers fielded AI models, without noticing biases in those models.

- Amazon had to scrap an automated recruiting tool as it was found to be biased against women.[S2]
- A widely used face recognition software was found to be biased against dark-skinned women[S3] and dark-skinned men.[S4]
- Google Translate, the most popular translation engine in the world, shows gender bias. "She is an engineer, he is a nurse," when translated into Turkish and then again into English, becomes "He is an engineer, she is a nurse."[S5]

For our purposes, the important point of the first Noble example is this: if software designers had been more intentional about soliciting feedback from the Black community, then they could have changed how check-ins are weighted in the overall Yelp rating system. As to the other examples, in each case, there was some discriminatory effect that was easy to detect and repair,[S6] but developers just failed to test for those biases.

There is a solution to all these problems: if a small group of people builds software for the larger community, they need to listen more to the concerns of the larger community. For that to work, the smaller group of developers has to admit the larger group into their design processes—either via a) changing the reward structures such that there are inducements for the few to listen to the many (for example, by better government legislation or professional standards); b) inclusion practices that admit the broader community into the developer community; or c) review practices where the developers can take better and faster feedback from the community. To say that another way, from an ethical perspective, it is good practice to give software to someone else and let them try to break it. For a discussion on tools to discuss that process, see the TESTED toolkit (discussed at the end of this article).

References

S1. S. U. Noble, *Algorithms of Oppression*. New York, NY, USA: New York Univ. Press, 2018.

S2. J. Dastin, "Amazon scraps secret AI recruiting tool that showed bias against women," *Reuters*, Oct. 2018. [Online]. Available: https://reut.rs/2Od9fPr **<AU: Please provide the volume number and the page range.>**

S3. L. Hardesty, "Study finds gender and skin-type bias in commercial artificial-intelligence systems," Massachusetts Inst. Technol., Cambridge, MA, USA, Feb. 2018. [Online]. Available: https://news.mit.edu/2018/study-finds-gender-skin-type-bias-artificial-intelligence-systems-0212

S4. "Wrongfully accused by an algorithm," *New York Times*, Jun. 2020. [Online]. Available: https://www.nytimes.com/2020/06/24/technology/facial-recognition-arrest.html **<AU: Please provide the volume number and the page range.>**

S5. A. Caliskan, J. J. Bryson, and A. Narayanan, "Semantics derived automatically from language corpora contain human-like biases," *Science*, vol. 356, no. 6334, pp. 183–186, Apr. 2017. [Online]. Available: https://science.sciencemag.org/content/356/6334/183, doi: 10.1126/science.aal4230.

S6. J. Chakraborty, S. Majumder, and T. Menzies, "Bias in machine learning software: Why? How? What to do?" in *Proc. 29th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, 2021, pp. 429–440, doi: 10.1145/3468264.3468537.

(or as a cursory add-on to) the rest of the curriculum (often in some separate subject called "Ethics in Computer Science" or an all-too-short module within a longer course). We believe there is more value in an integrated approach where ethics is woven into every subject and material from one subject informs the ethical discussions of another. Perhaps, more substantively, we believe our profession should acknowledge, embrace, and act to ensure that an engineering mindset is an ethical mindset.

OTHER WORK ON INTEGRATING ETHICS ACROSS THE COMPUTER SCIENCE CURRICULUM

We are not the only ones to discuss an integrated approach to ethics across the computer science (CS) curriculum.[S7,S8] **<AU: References must be cited in numerical order as per magazine style. Sidebar References [3, 9] were changed to [S7] and [S8], and the remaining references were renumbered accordingly. Please check throughout.>** A motivation present in the CS ethics curriculum is to contextualize technology by connecting it to its societal impact. Krakowski et al.[S9] referred to this coupling as a "sociotechnical" curriculum and highlight the curriculum's success, which corroborates with previous case studies, including a survey reviewing 115 university tech ethics course syllabi.[S7,S10] Fiesler et al.[S7] note the importance of interdisciplinary learning and the challenge to standardize teaching while considering the many variations of "tech ethics" integrations in present curricula.

Prior researchers have argued that ethics should be introduced as early as Programming 101 to prevent the "I'm just an engineer" mindset and underscored the need for more interdisciplinary collaboration with domains like philosophy to create a standard infrastructure for teaching and embedding ethics across CS. A European study[S11] **<AU: Please confirm the citation of the original reference [5] (Stavrakakis et al.) as the new reference [S11] here.>** on the importance of CS ethics curriculum coincides with previous research on 1) the widespread embrace for ethics curriculum integration; 2) the lack of hours dedicated to ethics teachings; and 3) the previous misconceptions of ethics referencing it as a standalone topic or area of concentration rather than a foundational concept present in main domains such as artificial intelligence, data science, and security. In accordance with the present results, studies have detailed the current "ethics crisis" and present a call for computer scientists to make the same strides as climate scientists by utilizing collaborative teaching practices and exchanges of knowledge.[S7,S12]

The suggestions for future work involve using ethics as a pedagogical lens such as "responsible" versus "irresponsible computing" frameworks[S13] and assessing academics' levels of ethical awareness to improve curriculum integration by beginning with educating professors.[S14] A challenge for future research is to expand the ethics theory at present and connect tech ethics pedagogies to feminist theory and critical inquiry. Williams et al.[S15] spotlight the dangers of minimizing ethics into "consequentialist, duty, or virtue ethics" and argue that students are not prepared for real-world scenarios involving structural societal systems of inequity. The current state of the literature emphasizes an ethics crisis but an open community of academics willing to learn how to successfully integrate tech ethics given the right curriculum.

References

S7. C. Fiesler, N. Garrett, and N. Beard, "What do we teach when we teach tech ethics?: A syllabi analysis," in *Proc. 51st ACM Tech. Symp. Comput. Sci. Educ.*, Feb. 2020, pp. 289–295, doi: 10.1145/3328778.3366825.

S8. B. J. Grosz et al., "Embedded EthiCS: Integrating ethics across CS education," *Commun. ACM*, vol. 62, no. 8, pp. 54–61, Aug. 2019, doi: 10.1145/3330794.

S9. A. Krakowski, E. Greenwald, T. Hurt, B. Nonnecke, and M. Cannady, "Authentic integration of ethics and AI through sociotechnical, problem-based learning," in *Proc. 12th AAAI Symp. Educ. Adv. Artif. Intell.*, Jan. 2022, pp. 12,774–12,782, doi: 10.1609/aaai.v36i11.21556.

S10. B. S. Baumer, R. L. Garcia, A. Y. Kim, K. M. Kinnaird, and M. Q. Ott, "Integrating data science ethics into an undergraduate major: A case study," *J. Statist. Data Sci. Educ.*, vol. 30, no. 1, pp. 15–28, Mar. 2022, doi: 10.1080/26939169.2022.2038041.

S11. I. Stavrakakis et al., "The teaching of computer ethics on computer science and related degree programmes. A European survey," *Int. J. Ethics Educ.*, vol. 7, no. 1, pp. 101–129, Apr. 2022, doi: 10.1007/s40889-021-00135-1.

S12. I. D. Raji, M. K. Scheuerman, and R. Amironesei, "You can't sit with us: Exclusionary pedagogy in AI ethics education," in *Proc. ACM Conf. Fairness, Accountability, Transparency*, Mar. 2021, pp. 515–525, doi: 10.1145/3442188.3445914.

S13. L. Cohen, H. Precel, H. Triedman, and K. Fisler, "A new model for weaving responsible computing into courses across the CS curriculum," in *Proc. 52nd ACM Tech. Symp. Comput. Sci. Educ.*, Mar. 2021, pp. 858–864, doi: 10.1145/3408877.3432456.

S14. R. T. Hans, S. M. Marebane, and J. Coosner, "Computing academics' perceived level of awareness and exposure to software engineering code of ethics: A case study of a South African university of technology," *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 5, pp. 585–593, 2021, doi: 10.14569/IJACSA.2021.0120570.

S15. R. M. Williams, S. Smarr, D. Prioleau, and J. E. Gilbert, "Oh No, Not Another Trolley! On the need for a co-liberative consciousness in CS pedagogy," *IEEE Trans. Technol. Soc.*, vol. 3, no. 1, pp. 67–74, Mar. 2022, doi: 10.1109/TTS.2021.3084913.

To make this concrete, at the end of this article, we offer TESTED, an example of a rewrite of a graduate automated SE course. In that rewrite, issues of ethics and responsibility are "baked into" the whole subject (and are not some clumsy post hoc add-on). TESTED is a proof by example that we can achieve the ethical engineering mindset without detracting from the core technical topics of a subject.

Who Should Teach More Ethics?

Not a Problem for Software Engineers?

Some argue that the ethical issues of software are best left up to politicians (to make legislation) or lawyers (who can argue the nuances of that legislation in court). In reply, we say that the ethical problems raised by software, and in particular intelligent software, are now so complex and rapidly evolving that lawyers and politicians just cannot keep up. At least some of these ethical issues must now be measured and mitigated by the teams designing and maintaining those systems.

Further, addressing the ethical issues posed by software should not be something that engineers do out of statutory obligation but out of a sense of obligation to our fellow human beings. We should be self-policing our practices and not outsourcing requirements to those without the technical skills of our profession. That being said, often, powerful economic, political, or social forces can (inadvertently) incentivize unethical behavior. As such, top-down governance via legislation or policy can play a role in setting the occasion to drive changes in ethical engineering practice. However, bottom-up cultural shifts can drive change more efficiently and enhance those statutory efforts (should they occur).

In our view, issues of, for example, software bias in intelligent systems are now such a pressing matter that it is reckless and irresponsible to assume some other community will fix it. This is an "all hands on deck" situation where many communities need to offer their skills to address a pressing social problem. And there is much the SE community can offer for this problem. Berk et al.[5] famously said in 2017 that "It is impossible to achieve fairness and high performance simultaneously (except in trivial cases)." But in 2021, an SE research team showed that Berk et al. were wrong since many of the things done to fix fairness are also the sampling operators widely used in software analytics to improve prediction. Hence, Chakraborty et al.[S6] **<AU: Several references were duplicated in the reference list ([6] [S6]) and ([8], [S1]). Because they were cited first in the sidebars, the citations of [6] and [8] were changed to citations of [S6] and [S1] in the text, and the duplicates were deleted from the main reference list. References must be cited in numerical order as per magazine style, so and the remaining references were renumbered accordingly. Please confirm that they are adjusted correctly.>** showed that their Fair-SMOTE system could improve not only fairness measures but also predictive performance.

Not a Problem for Technologists?

Some say software is a technology and, as such, is selected to favor the ruling elite.[6] In this view, software is as inherently bad (racist, sexist, misinforming) as anything else selected by their social context. Hence, in that view, there is no value in fixing, for example, algorithms until we first fix the society that selects and deploys them. Noble,[S1] for example, wants "decoupling of advertising and commercial interests from the ability to access high-quality information on the Internet"; to "suspend the circulation of racist and sexist material that is used to erode our civil and human rights"; and to require that all search results be annotated to symbolize, for example, pornography (in red); business or commercial material (in green); entertainment (in orange), etc.

But is this viewpoint incomplete? Does it underestimate the number of choices within a technology (most of which are unexplored)? Not all technology inexorably returns a single output hardwired into its design. Some algorithms are exploratory tools that help humans trade off between numerous competing goals. Other software has a large set of configuration parameters that can change all manner of things, including the false positive rate between different populations. Just as algorithm designers need to know more about the broader social issues of their work, so too do social theorists need to know about algorithms. In this way, these groups can better work together while discussing, for example, how language can marginalize and disfranchise social groups[7] or how the Zitzler predicate can sort items on the Pareto frontier during multiobjective optimization.[8]

What we seek is a "two-way street" between what we might call the humanities view (which is light on CS knowledge) and the CS view (which is light on knowledge of the broader social context). For example, Gebru[9] wants regulation that "specifically says that corporations need to show that their technologies

are not harmful before they deploy them." Implementing that requirement, at a company-level scale, would require many things, including the fairness testing methods discussed here.

How to Teach Ethics (an Integrated Approach)

To repeat the main theme of this article: We believe it is important to integrate concepts related to ethical decision making throughout the various courses in CS curricula. More specifically, we say that every course in the CS curriculum should explore the following questions:

- Who could get hurt by the software from that subject?
- How can that hurt be mitigated? Here, students will be asked to take methods from one subject and apply them to another.
- How can that kind of software be changed to empower more people such that this kind of hurt does not happen in the future?

By answering these questions, students will be encouraged to consider who is empowered, or disempowered, by technology and the decisions they make when developing and deploying that technology. For example, students should be taught it's critical

- for someone else to be able to review and understand what you are doing
- for others to have opportunities to object to all or part of what you do
- to respond to any objections raised (perhaps by changing what you are doing).

In this view, students in, for example, a database course may be led to critically analyze how well a database supports the following:

- privacy (that is, controlling who can see what about whom)
- forgetting (that is, whether all the details about one person can be removed from the systems)
- access (that is, whether individuals can check who has seen what about their records)
- correction (for example, if a government database has some error about someone, whether someone can repair that error).

For another example, consider our TESTED data science class. For this article, the important point of TESTED is that it shows that ethical issues (specifically, issues of accountability) can augment and improve syllabus design.

TESTED is a set of coding assignments, written in Lua (which is a small and simple Python-like language but with far less overhead). Students use these samples as an executable specification that they must reproduce in any other language they like (except Lua). Each of these assignments is about one to two weeks of work. Hence, it is suitable for homework or (by combining several modules) a large end-of-term project.

Given all the examples in "On the Need for Ethical Software," testing is an essential component of AI. Hence, TESTED is very focused on test-driven development—by both developers and outside groups. TESTED assumes that the best way to test something is to give it to someone else and watch them break it. This is actually a core principle of ethical programming. Vance et al.[10] argue that a precondition for accountability is the knowledge of an external audience, who could approve or disapprove of a system. Hence, TESTED includes five accountability-support tools.

1. operators for learning the boundaries of a system's competency
2. methods for looking beyond those boundaries (taken from cognitive psychology)
3. human-readable model generation methods that can extract symbolic descriptions from training data (since that is what humans need for explaining a system)
4. cost-effective sampling methods that let outsiders probe a system, looking for interesting (or alarming) behavior
5. semisupervised learners where algorithms make conclusions based on a small sample of the total data space (so humans are not overwhelmed with excessive questions).

TESTED encourages a mix-and-match approach to AI (where developers can exert much control over what functionality they deliver). To foster that approach, students are encouraged to reflect on all the overlap within the previously mentioned accountability-support tools.

- To explore beyond the boundaries of the current system, TESTED uses hierarchical clustering to implement the tautology and instance selection hierarchies used in repertory grids (see Figure 1). Repertory grids are a tool proposed by the cognitive psychologist George Kelly as a method for eliciting tacit knowledge. Niu and Easterbrook[11] **<AU: The Niu and Easterbrook reference was**

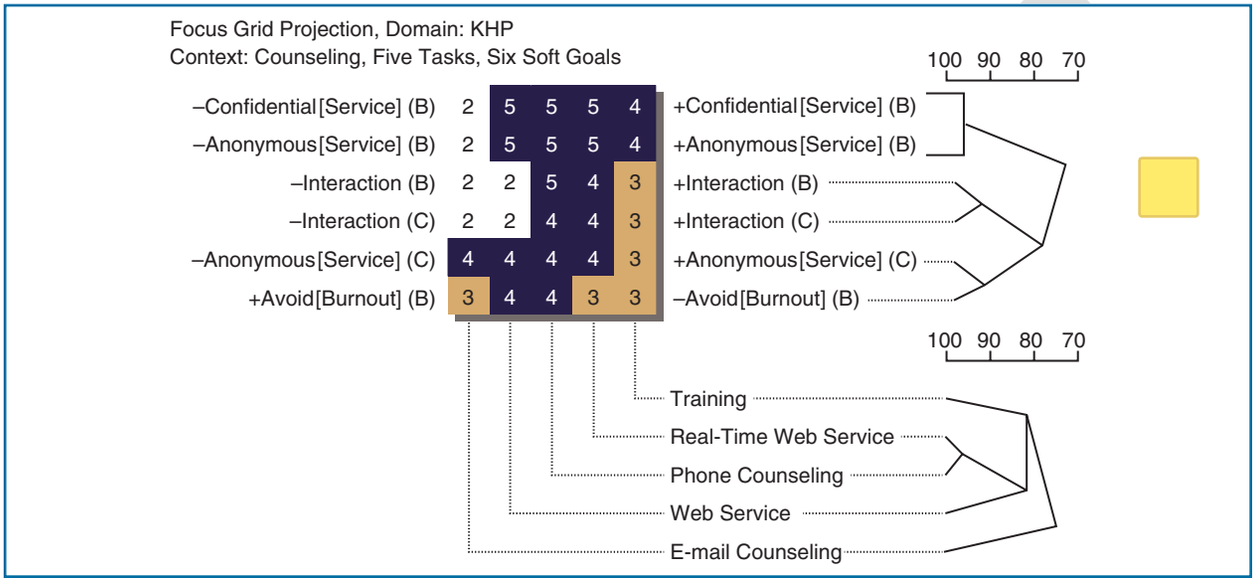

FIGURE 1. An intro TESTED homework: implement the row/column clustering of a repertory grid. ***<AU: Kindly spell out KHP so that the definition can be added to the caption.>*** (Source: Taken from Niu and Easterbrook.[11])

added to the end of the reference list as the new Reference [11].> comment that repertory grids are widely recognized as a domain-independent method for externalizing individuals' personal constructs. Interviewees are invited to offer their own examples from their own domain. Then, they are asked: "Given three examples (picked at random), on what dimension is one example most different from the other two?"

- To define a system's boundary, TESTED reuses the same clustering tools applied to build the repertory grids. To check if new inputs fall within the data used to train a system, TESTED runs the new examples down its tree of clusters. New inputs are anomalous if they fall far from the median of a cluster. An anomalous example can either:
- trigger an alert based on **<AU: Please check that the change from "trigger an alert that" to "trigger an alert based on" is correct.>** any conclusion for this example (since it is based on information that is out of the scope of the training set)
- trigger model updates (and, to keep that tractable, those updates can be restricted to just the clusters suffering from anomalies).

Further to the aforementioned, there are many other ways TESTED can show that, under the hood, there is much commonality in many AI systems. For example, once we can recursively cluster data (using the methods described previously), then there are many ways to use the leaf clusters found by that recursion.

- Nearest-neighbor algorithms can be very slow. They can be sped up by looking only for near neighbors within the same leaf cluster.
- By sampling and labeling only a few examples per leaf, then our tree of clusters becomes a semisupervised learner. In this approach, all examples of a leaf share the labels seen on one (or slightly more than one) example per leaf. Semisupervised learners are very important for human-in-the-loop systems since they mean humans are pestered for their opinion on only a small subset of the data.
- By comparing the labels collected on different branches, it is possible to compute actions that adjust results from one leaf L1 to another leaf L2. Note that if the comparison is based on some multiobjective criteria, then this whole approach converts from "classification" and "regression" to a "multiobjective optimizer." Note also that by discretizing numeric ranges such that we use only ranges with different ratios of examples from L1:L2, we can then "bunch up" many ranges into far fewer ranges (which makes rule or decision tree generation much simpler and faster).

ABOUT THE AUTHORS

TIM MENZIES is with North Carolina State University, Raleigh, NC USA. **<AU: Please supply the current position title and the postal code of the current affiliation.>** Contact him at timm@ieee.org.

BRITTANY JOHNSON-MATTHEWS is with George Mason University, Fairfax, VA USA. **<AU: Please supply the current position title and the postal code of the current affiliation. Also, please provide a high-resolution author photo.>** Contact her at johnsonb@gmu.edu.

DAVID L. ROBERTS is with North Carolina State University, Raleigh, NC USA. **<AU: Please supply the current position title and the postal code of the current affiliation. Also, please provide a high-resolution author photo.>** Contact him at dlrober4@ncsu.edu.

LAUREN ALVAREZ is with North Carolina State University, Raleigh, NC USA. **<AU: Please supply the current position title and the postal code of the current affiliation. Also, please provide a high-resolution author photo.>** Contact her at lalvare@ncsu.edu.

- Once the leaves are sorted, then those actions can be reported to the members as either
  - plans on how to improve things (that is, how to change a conclusion from a worse leaf to a better leaf)
  - monitors of what conditions can change conclusions from a better to a worse leaf.

Returning now to the issue of ethics and accountability (and the need for outsiders to test a system), the previously mentioned description of TESTED simplifies the interactions between human and AI systems. Semisupervised learners reduce the number of interactions required between humans and AI. Visualizations such as Figure 1 offer high-level symbolic descriptions needed for human comprehension (and for a more detailed view, we can use decision trees). Anomaly detectors tell us when a current model needs to be extended. Repertory grids let us explore the space of concepts important to users. Since our RepGrid analysis scales from small examples (like Figure 1) to much larger datasets, we can run the same attribute/example clustering on the concepts of the users.

The Engineering Mindset Can Be an Ethical Mindset

Our challenge in building a professional culture comprising an ethical mindset is not to appropriate the success of technical fields but to replicate and adapt for our purposes. Technology rarely, if ever, exists for technology's sake. The artifacts engineers produce exist in a context comprising the environment and people that interact with the artifact or are impacted by it. These artifacts aren't purely technical, they're sociotechnical, and traditional engineering education often eschews the socio in favor of the technical.

In this short article, we have offered motivation and an example of why and how ethics can be an integral part of contemporary CS curricula. Bringing the socio and technical closer to parity of focus in engineering education doesn't represent a monumental change in the skills we teach; it simply requires that ethical considerations be taught to be as natural a part of the solution to an engineering problem as the choice of compiler, programming language, dataset, etc. in SE.

The tools of other communities may provide inspiration, but we as a community of engineering professionals already have a well-stocked toolbox from which we can build a broad solution to training ethically minded future engineers. The engineering mindset is often described as comprising, in large part, "systems thinking" or the idea that the world around us is linked in (often) subtle but critical ways. Systems thinking promotes maintaining awareness of these relationships to identify structures and to be conscious of how

choices interact with those structures. In essence, the engineering mindset can be thought of as a dedication to asking the right questions, at the right times, and taking into account the right relationships. This is not at all different from the ethical mindset, where asking the right questions, at the right time, and accounting for the right people are fundamental.

In short, engineers trained to have an engineering mindset already have the skills to acquire an ethical mindset—they need the context, motivation, and experience to engage in these systems thinking tasks, keeping the people who are a part of the structure of these systems on par with the technical. Ethical frameworks, like TESTED, can be a powerful tool in instilling broad adoption of the ethical mindset when deployed widely in education.

References

1. "Developer survey results 2018." Stack Overflow. [Online]. Available: https://insights.stackoverflow.com/survey/2018 **<AU: Please provide the date of access.>**
2. "Geoethics: Ethical decision-making." SERC. [Online]. Available: https://serc.carleton.edu/geoethics/Decision-Making **<AU: Please provide the date of access.>**
3. "Resources." Online Ethics Center. [Online]. Available: https://onlineethics.org/resources?combine=software&field_keywords_target_id=&field_resource_type_target_id=13236 **<AU: Please provide the date of access.>**
4. E. M. Peck. "Integrating social responsibility into core CS." GitHub. [Online]. Available: https://evanpeck.github.io/projects/responsibleCS **<AU: Please provide the date of access.>**
5. R. Berk, H. Heidari, S. Jabbari, M. Kearns, and A. Roth, "Fairness in criminal justice risk assessments: The state of the art," 2017, *arXiv:1703.09207v1*.
6. D. F. Noble, *America by Design*. New York, NY, USA: Knopf, 1977.
7. P. Mpofu and A. Salawu, "Linguistic disenfranchisement, minority resistance and language revitalisation: The contributions of ethnolinguistic online communities in Zimbabwe," *Cogent Arts Humanities*, vol. 5, no. 1, Nov. 2018, Art. no. 1551764, doi: 10.1080/23311983.2018.1551764.
8. E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in Proc. Int. Conf. Parallel Probl. Solving Nature, Springer-Verlag, 2004, pp. 832–842, doi: 10.1007/978-3-540-30217-9_84.
9. K. Adams. "Timnit Gebru envisions a future for smart, ethical AI." Marketplace. [Online]. Available: https://www.marketplace.org/shows/marketplace-tech/timnit-gebru-envisions-a-future-for-smart-ethical-ai/ **<AU: Please provide the date of access.>**
10. A. Vance, P. B. Lowry, and D. Eggett, "Increasing accountability through user-interface design artifacts: A new approach to addressing the problem of access-policy violations," *MIS Quart.*, vol. 39, no. 2, pp. 345–366, Jun. 2015, doi: 10.25300/MISQ/2015/39.2.04.
11. N. Niu and S. Easterbrook, "So, you think you know others' goals? A repertory grid study," *IEEE Softw.*, vol. 24, no. 2, pp. 53–61, Mar./Apr. 2007, doi: 10.1109/MS.2007.52.