STM JOURNALS

Journal of
Software Engineering Tools & Technology Trends

ISSN: 2394-7292
Volume 12, Issue 1, 2025
January–April
DOI (Journal): 10.37591/JoSETTT

https://journals.stmjournals.com/josettt

Review JoSETTT

How Ethics Impact Software Development

V. Basil Hans*

Abstract

Software development shapes modern society in the age of rapid technological advancement. However, possessing great power also brings the obligation to incorporate ethical practices throughout the development process. Fairness, transparency, privacy, inclusivity, and accountability are key to ethical software development. This article discusses ethical software development principles like user-centric design, accessibility, algorithm bias reduction, data privacy, and security. It examines ethical issues developers face, such as artificial intelligence (AI) misuse, discriminatory algorithmic results, and privacy breaches. It also emphasizes the importance of ethical cultures in development teams and organizations, using the ACM Code of Ethics and IEEE Ethically Aligned Design principles. The article also emphasizes the need for proactive measures, continuous education, and ethical governance to address AI and big data challenges. Software developers can build trust, reduce harm, and ensure technological innovation benefits society by promoting ethical practices. In the modern era of technological advancement, software development plays a pivotal role in shaping societies and influencing global progress. However, the ethical dimensions of software engineering are often overlooked in the race to innovate. This article explores the critical intersection of ethics and software development, emphasizing its impact on decision-making, user trust, and societal well-being. Ethical considerations influence various aspects of the development lifecycle, including algorithm design, data privacy, accessibility, bias mitigation, and sustainability. Neglecting ethical principles can lead to unintended harm, such as discriminatory algorithms, security vulnerabilities, or breaches of user confidentiality. Conversely, embedding ethics into development processes fosters transparency, inclusivity, and long-term success. This article emphasizes the significance of incorporating ethical frameworks into software engineering education, workplace policies, and team interactions, drawing insights from case studies and best practices. Ultimately, it argues that ethical software development is not merely a moral obligation but a strategic imperative in building a fair, equitable, and sustainable digital future.

Keywords: Software, ethics, shared responsibility, data protection, developer, user

INTRODUCTION TO ETHICAL SOFTWARE DEVELOPMENT

Software is everywhere, so ethics are becoming important in software development. Software developers and companies shape technology ethics, so they must consider the common good and harm. As software becomes more pervasive in the world, issues such as user trust, social responsibility, and ethical dilemmas surrounding the development and release of software are becoming more important. It is important to consider ethical principles throughout the whole lifecycle of the development and use of software – from before the creation of a project, to the implementation, release, updates, and end of the software.

***Author for Correspondence**
V. Basil Hans
E-mail: vhans2011@gmail.com

Research Professor, Department of Management & Commerce, Srinivas University, Mangaluru, Karnataka, India

Received Date: January 16, 2025
Accepted Date: January 22, 2025
Published Date: March 18, 2025

Citation: V. Basil Hans. How Ethics Impact Software Development. Journal of Software Engineering Tools & Technology Trends. 2025; 12(1): 29–35p.

Discussing the ethical implications of software development lends itself to failure on two rhetorical

fronts. If the topic is discussed broadly, the complexity and breadth of the issues is glossed over. If the discussion is narrow and focused, it is easy to neglect the many circumstances that comprise the production and use of software and to quickly lose the audience. An assessment of ethical software development requires to treat software as simply one little gear in a massive social, political, and economic machine, while also acknowledging that the machines which software currently powers are diversified and work in unusual and undiscovered places. This justifies a multi-faceted approach to be taken in the issue of ethics in software development [1].

The conversation will almost automatically rely on ethical considerations, categorized by the agents that are responsible at each stage in the software lifecycle, although overlooking ethical factors as they mix and conflict is a tough weight to manage. It would be unjustified to hate any reader with a preoccupation that, while essential to ethical software development, may seem to underplay its fundamental importance.

LITERATURE REVIEW

Software engineers shall commit themselves to making the analysis, specification, design, development, testing and maintenance of software a valuable and respected career. In accordance with their dedication to the health, safety and welfare of the public, software engineers shall adhere to the following eight principles:

1. Public – Software developers shall act consistently with the public interest.
2. Client and Employer – Software engineers shall act in a manner that is in the best interests of their client and employer compatible with the public interest.
3. Product – Software developers shall ensure that their products and related revisions meet the highest professional standards imaginable.
4. Judgment – Software engineers shall retain integrity and independence in their professional judgment.
5. Software engineering managers and executives must accept and encourage ethical software development and maintenance management.
6. Software engineers must promote the profession in the public interest.
7. Software engineers must be fair and supportive of coworkers.
8. Software engineers are expected to engage in lifelong learning and promote ethical practices [2].

Software development ethics are crucial. Software engineers must be ethical and professional. Engineers must preserve social principles and build user trust. They must consider both the planned and unintended effects of their job.

Software developers must maintain integrity under tight timelines and competitive constraints. Responsible software engineers follow ethics, unlike those who prioritize speed above accountability [3].

ETHICAL SOFTWARE DEVELOPMENT PRINCIPLES

Introduction: Software Ethics

Software ethics examines how software affects society. Software developers and computer technology professionals must consider on their work's impact on society and their ethical obligations to mitigate harm. Software developers share responsibility for the inevitable and foreseeable effects of a software project, unlike physicians, who must “First, do no harm” when joining the field [4]. In general, software engineers must be more attentive and analytical as project risk increases.

The negative influence of software is mostly due to user unethical behavior. However, this does not mean software creators are not liable for those usage and their impact. The following principles drive ethical software development and encourage contemplation on software's values and ramifications. They start, but are not exhaustive. Modern civilization and software pose ethical issues that no one expected to write about. Further, what is ethically and socially accessible or even required in one field

(like medical software) may not apply in another (like entertainment software design). This is normal. Software practices are extensive because software is. Ethical reflection must be thorough. Does dishonest software affect more than its users, society, or the assumed victims of illegal or untimely access? Shared responsibility should be measured by risk, software social impact, and content. Finally, one should be committed to a comprehensive engagement with the probable harms and social impact of one's activities and to minimizing unethical impact while structuring software methods to achieve expected advantages (social, monetary, etc.).

Software Design and Development Ethics

Ethics-related non-functional requirements were clear throughout the software design and implementation of a broker-agnostic energy storage system connectivity solution. Software developers developing new software-intensive systems must model safety, security, and privacy non-functional requirements (NFRs). Battery energy storage systems (BESS) aim to provide a dependable and safe way to link a number of devices and technology to the community. Since the BESS is connected to the public power system via charge-discharge cycles with key performance indicators (KPIs), safety is a concern. When using such systems in major cities or vital industries, safety and security must be considered integrally linked. The conversation with C-level developers in the telecommunications sector focusing on operation, management, and control software (operations support systems [OSS] and business support systems [BSS]) for large multinational telecom operators and vendors shows that software companies view those systems as inherently safe, designed according to ethical and legal precepts, and meeting the latest standards. This system can characterize safety and security compliance in social responsibility and citizen trust [5]. Documenting system safety measures and unambiguous procedures for undesired scenarios will prevent software from acting unexpectedly.

Privacy/Data Protection

Data privacy and protection are essential to creating trustworthy software. Developers have an ethical obligation to protect user data from abuse or theft. Frameworks such as the General Data Protection Regulation emphasize this legal requirement. Data must be protected and acquired with comprehensive user consent. Like good design can inspire trust in ‘privacy by design’ concepts, poor design can demonstrate questionable design decisions. Data collection, content, and destination should be transparent in the app. These frameworks show that betrayal of this obligation can lead to software complaints and 'distrusting sabotage' in which users contribute false data to poison the system. This can include refusing to disclose useful or secret data or entering false or outdated data. User harm can escalate beyond data loss to identity theft, harassment campaigns, hostile social engineering, and other demonstrable harm. Therefore, software must work hard to keep user data anonymous and secure. Developers should also be aware of their legal obligations to collect data from users and act ethically to store only the minimum data needed for program functionality and respect privacy at all stages of software design, development, distribution, and use.

Transparency and Accountability

Software transparency helps prevent misuse and unethical use and builds confidence between developers and users [6]. Software transparency includes explicit explanation regarding functionality and data policies. An ethical software should be transparent. Developers must understand how their product affects users and stakeholders and be transparent about it. In developing any software program or system, transparency should be a top priority, and developers should expand transparency features whenever possible and beneficial.

The actor is accountable for their actions. Software developers should accept responsibility for the implications of their applications and ensure their ethical use. Therefore, accountable developers actively learn about the potential implications of misuse and are open to criticism and discussion with their colleagues, developers, and end users, making adjustments wherever possible. Accountable developers also seek third-party mechanisms like investigations, audits, mechanisms for reporting bugs

and unethical uses directly back to developers, and user feedback to help them reflect on their software. This means that developers and other developers contributing to a framework are both responsible for transparency and accountability. Larger, multi-actor systems make responsibility evading easy, so only their least accountable element is responsible.

While many ethical breaches in software are purposeful, malevolent software can be produced in obscurity. Many of the most criticized algorithms are constructed on sophisticated or unknown methodologies, not their core purpose. Parties knowledgeable of dangerous software practices generally struggle to implicate the developer. Even basic methods like making software free or obscuring its function can help committers and contributors escape responsibility for product misuse. Consumers and end users are unprotected against these methods unless they act for a single, well-publicized case, which is obviously underfunded and prohibitory for most.

Bias and Fairness

Fair and impartial software development is crucial. Bias can be introduced into today's software products, either intentionally or accidentally, resulting in unfavorable outcomes for user categories it was not planned to serve. Identifying biases in software or algorithms allows for active counteraction of unfairness. There should be a belief that discrepancies are not permanent and may be corrected during software development. Researchers and practitioners must actively pursue fair and unbiased ways to design and improve software and computational systems [7].

Research typically develops and discusses software solutions for de-bias analytics. Too often, conversations are seen as single-directional talks focused on providing insights for developers and researchers, with created solutions deemed analyst action. Biases are likely mentioned and defined in the publications. A more holistic strategy is needed to address these prejudices, which may explain why they are not challenged as thoroughly. Everyone in software development must work.

Fairness in software is inextricably linked to social justice and its implications. Modern outreach and computational systems shape activities in a digital culture, therefore those things can change the world, if not physically. Developers and researchers should challenge their own, their teams', and perceived global normative fairness standards in computational work. We must build software that is fair and impartial to all populations, both developers and users.

Accessibility

Accessibility is the process of creating and providing software and online apps for persons with different abilities and limitations. In other words, software versions and web apps produced and provided by software professionals should enable end-user engagement regardless of ability or disability. Given the preceding criteria, software products can be made accessible by implementing Web Content Accessibility Guidelines 2.0 (WCAG 2.0) or higher accessibility rules. WCAG compliance is required by many national laws. W3C and other accessibility standards like Section 508, European Accessibility Act (EAA), and Indian Standards (IS) 5568 in India require software firms to create accessible products [8]. Software must meet accessibility criteria to be useable by disabled people. Unfortunately, software professionals do not always follow accessibility requirements when developing programs. Waterfall or iterative and incremental software development is used. Waterfall works sequentially through requirements analysis, design, coding, testing, and maintenance. Software firms iterate or incrementally build products due to time-to-market constraints. To satisfy changing business needs, Agile and Scrum recommend iterative and incremental software development. Business requirements are elicited, analyzed, planned, coded, tested, and shipped faster in each iteration due to these models. These approaches don't handle software accessibility requirement analysis, design, or testing. Software developers check and validate products based on end-user functional requirements in the application domain. Such a gap exists in building accessible software products. Social and web application developers must fill the gap in software usability and accessibility.

Software Development Ethics Frameworks

Ethical decision-making frameworks offer two main models for assessing complicated ethical challenges software engineers may confront [9]. These tools encourage and nudge developers to evaluate actions and potential consequences from widely developed, executed artifacts and software development processes that may cause unintended, crucial, or severe harm to humans, societies, nations, and the environment. This examination can be viewed from different but complimentary angles, including important values and uncertain ethical judgments. Consider the probable effects of accessible actions on the damage ends stated above. Considering the well-being of affected stakeholders can help make significant contemplation as comprehensive as possible without complicating it. Interpret their major interests and draw up m(any) ambitious hypotheses about the extent of (negative) outcomes that are possible within the scope. These interests should cover health, economic achievement, and specific businesses or families.

Proponents of the models say integrated application with multiple software development teams efficiently represents the different interests and financial rewards from science-market cooperation. First, a qualitative illustration of how these frameworks were used to digital platform giants is offered to demonstrate their applicability. After that, the models are assessed for their capacity to promote ethical thinking and their pros and cons. These technologies are said to help software developers make ethical decisions in both specific situations and business strategies.

Ethical Software Development Case Studies and Best Practices

Software affects almost every element of modern life. As organizations incorporate software safety study results into design, utilitarianism might motivate them to create better, more dependable software. Today, software is used in practically every country and touches almost every facet of daily life. Software is now considered “nearly as indispensable as water, food, and shelter.” Software-based systems handle security, transport, finance, commerce, and even entertainment. Software quality is an important and often-discussed problem. These vital systems prioritize reliability, trustworthiness, and safety. Safety and hazard analysis for safety-critical systems, a subset of software-based systems, has been the subject of much research and testing, but the results have been mixed due to the long and difficult development process.

Using those analyses may also improve compliance and prevent lawsuits. Finally, these results imply that software decision makers can improve the industry's public image by following ethical norms. Industry could improve software and services by incorporating software safety study results into design [10]. This feedback would improve industrial products and software-based system safety and reliability. This would follow utilitarianism: better, more dependable software benefits the most people. Safety research studies in the aviation industry should lead to better software over time. Through software ethical guidelines, industry may embrace better practices and be more responsible.

RESULTS

Fairness, transparency, accountability, and social well-being are the goals of ethical software development. Some typical findings and critical insights from our ethical software development study:

Knowledge of Ethics

Numerous developers lack proper ethical decision-making training.

- *Insight:* Developers often fail to spot ethical issues in their work, causing unintended effects.
- *Advice:* Integrating ethics into education and professional growth raises awareness.

Conflicts Between Business Goals and Ethics

- *Findings:* Deadlines, cost reduction, and profit typically trump ethics.
- *Insight:* Developers may release incomplete or biased software.
- *Advice:* Clear ethical norms in organizations help reduce these conflicts.

Fairness and bias

- Research shows that training data, design decisions, and implicit developer biases affect software systems.
- *Insight:* Unchecked biases in artificial intelligence hiring, lending, and police systems can perpetuate discrimination.
- Results can be more egalitarian with regular bias and fairness checks during development.

Gaps in Accountability

- In teams, ethical responsibility is often unclear.
- Ethical issues may be neglected or deprioritized without clear ownership.
- *Advice:* Appointing ethics champions or committees ensures responsibility.

Clearness and Explanation

- User understanding of decision-making is challenging in many systems due to lack of openness.
- This erodes trust, especially in high-stakes fields like healthcare and finance.
- *Recommendations:* Explainability throughout development boosts user confidence and regulatory compliance.

Including Different Views

- Diversity-free teams may miss ethical issues affecting marginalized groups.
- *Insight:* Diverse teams make inclusive software.
- *Recommendations:* Promoting team diversity and stakeholder involvement helps improve ethics.

Effects of Regulation

- Findings: GDPR (General Data Protection Regulation) has encouraged better data practices but is typically seen as reactionary.
- Insight: Ethical software requires more than compliance.
- Developers should exceed legal requirements with proactive ethics.

Tools, Frameworks

- *Results:* Ethics and decision-making frameworks are underused.
- Accessible tools help developers navigate complex ethical challenges.
- *Recommended:* Organizations should include ethical checklists, scenario analysis tools, and decision-making frameworks.

Long-Term Effects and Sustainability

- *Results:* Short-term thinking commonly guides software decisions, ignoring long-term user and environmental impact.
- *Insight:* Ethical software development anticipates and mitigates long-term risks.
- Sustainability can be improved by including lifecycle impact studies into development processes.

CONCLUSION

Technology has made ethical software development a need. Developers must consider ethics as software affects every aspect of society, from individual decisions to global systems. Ethical software development presents significant obstacles and opportunities, as this article has shown.

Key findings show awareness, responsibility, and inclusion gaps and business aims versus ethical imperatives. Ethical education, fairness and transparency in design processes, diversity in teams, and proactive involvement with social impacts are needed to address these difficulties.

The aim of ethical software development is to minimize harm while enhancing the well-being of individuals and society. Developers and organizations can design fair, transparent, and sustainable

software by using ethical frameworks and tools, creating a more egalitarian and trustworthy digital future.

REFERENCES

1. Wright DR. Motivation, design, and ubiquity: a discussion of research ethics and computer science. arXiv preprint. arXiv:0706.0484. June 4, 2007.
2. Gotterbarn D, Miller K, Rogerson S. Software engineering code of ethics is approved. Commun ACM. 1999; 42 (10): 102–107.
3. X-Team. Ethical issues in software development. [Online]. X-team.com. 2024. Available at https://x-team.com/magazine/ethical-issues-in-software-development
4. Wolf MJ, Miller KW, Grodzinsky FS. On the responsibility for uses of downstream software. Computer Ethics – Philos Enquiry Proc. 2019; 2019 (1): 3.
5. Cysneiros LM, do Prado Leite JC. Non-functional requirements orienting the development of socially responsible software. In: Nurcan S, Reinhartz-Berger I, Soffer P, Zdravkovic J, editors. Enterprise, Business-Process and Information Systems Modeling: 21st International Conference, BPMDS 2020. Cham, Switzerland: Springer International Publishing; 2020. pp. 335–342.
6. Obie HO, Ukwella J, Madampe K, Grundy J, Shahin M. Towards an understanding of developers' perceptions of transparency in software development: a preliminary study. In: 2023 38th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW), Luxembourg, September 11–15, 2023. pp. 40–45.
7. de Souza Santos R, Fronchetti F, Freire S, Spinola R. Software fairness debt. arXiv e-prints. arXiv:2405.02490. May 3, 2024.
8. Joshi S. Exploring the need of accessibility education in the software industry: insights from a survey of software professionals in India. arXiv preprint. arXiv:2401.00451. December 31, 2023.
9. Vakkuri V, Kemell KK, Abrahamsson P. Ethically aligned design: an empirical evaluation of the resolved-strategy in software and systems development context. In: 2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Kallithea, Greece, August 28–30, 2019. pp. 46–50.
10. Vinson NG, Singer J. A practical guide to ethical research involving humans. In: Shull F, Singer J, Sjøberg DIK, editors. Guide to Advanced Empirical Software Engineering. London, UK: Springer; 2008. pp. 229–256.

