On Integrating Ethicality in User Stories

Pankaj Kamthan
Computer Science and Software Engineering
Concordia University
Montreal, Canada
pankaj.kamthan@concordia.ca

Nazlie Shahmir
Information Services Governance
Canadian Pacific Railway Limited
Calgary, Canada
nazlie_shahmir@cpr.ca

***Abstract*—In recent years, software has increasingly become anthropomorphic, even autocratic. For example, software is being used exclusively for activities, such as decision-making, question-answering, or recommending, that in the past were either partly or entirely human. This has only contributed to the enduring issue of software ethics. In that regard, this paper models ethicality as a meta-quality attribute and proposes an ethically-sensitive, standards-based, technology-and-tool-independent, semi-formal framework, comprising interrelated conceptual (meta-)models that provide an understanding of ethicality, user story environment, and user story process. It outlines an approach of integrating ethicality naturally and systematically in the user story process, and provides illustrative and representative examples in support of this approach. Finally, it presents the results of a preliminary survey of students and professionals on their knowledge and experience of ethics in (agile) software projects.**

Keywords-axiology; conceptual modeling; ethical dilemma; human-centered agile methodology; interactive system; software quality

I. INTRODUCTION

The history of computer ethics predates that of software engineering [1, 2]. Fig. 1 gives an approximate timeline of different types of ethics that have been a subject of attention in relation to computing, as the role of computing itself evolved based on the needs of the society. However, increasing 'softwareization' of a variety of application domains, along with essentially uncontrolled and unlimited *malleability* of software, including those for unscrupulous or maleficent purposes, has made the issue of software ethics as exigent as ever. The consequences have ranged from innocuous and reversible, albeit at the cost of time and effort, to extremely nocuous and irreversible [3]. This situation is clearly untenable.

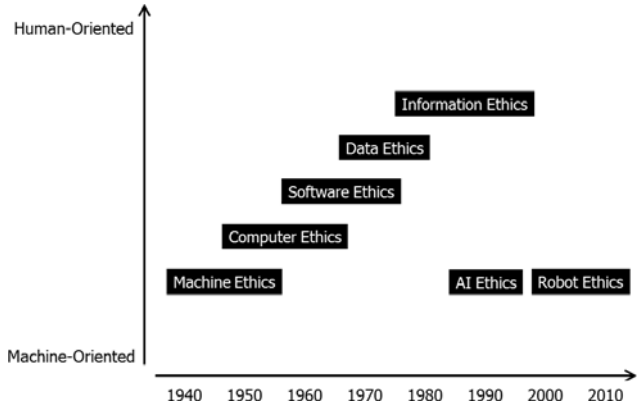

Figure 1. A panorama of different types of ethics over different decades.

It is known that requirements engineering (RE) is one of the earliest and most crucial phases in software development, primarily because of the tremendous responsibility it places on the stakeholders, and the considerable control they can exercise during this phase. The user stories are being deployed increasingly in recent years in agile software projects that adopt a *scenario-oriented* requirements engineering (RE) approach [4]. Therefore, the interest in this paper is providing the necessary basis for a conceptual framework for integrating ethicality naturally and systematically in the user story process.

The rest of the paper is organized as follows. In Section II, necessary background is provided and related work is discussed. The specifics of the construction of elements of the framework, along with rationale and description are presented in Section III. In Section IV, directions for future research are outlined. Finally, in Section V, concluding remarks are given.

II. BACKGROUND AND RELATED WORK

A. Nature of Ethicality from a Software Engineering Perspective

This paper distinguishes among actions or inactions that are *prudential*, *ethical*, and *legal* [5]. Fig. 2 illustrates the interrelationship between these concepts by means of a Venn Diagram. An action or inaction is prudential if it is in a person's interest; an action or inaction is legal if it is not explicitly prohibited by law of a jurisdiction; and an action or inaction is ethical if it does not violate certain (personal, organizational, and/or societal) codes of ethics. Furthermore, ethics could be either *deontological* (an action or inaction is ethical or unethical in itself) or *teleological* (an action or inaction is ethical or unethical depending on its consequences).

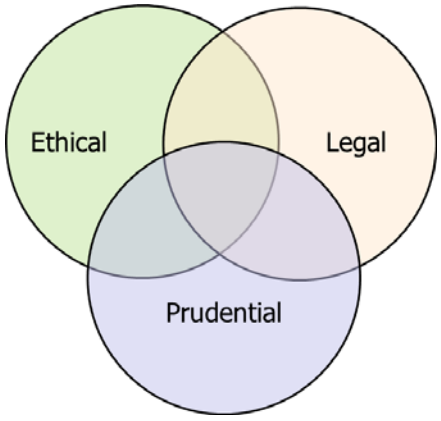

Figure 2. The ethical, legal, or prudential actions or inactions in context.

DOI reference number: 10.18293/SEKE2021-001

For example, while booking a flight it may be prudential for a traveler to have travel insurance, but it is teleologically unethical for a flight reservation system to add the cost of travel insurance to the cost of the flight automatically without the traveler's consent, and it may be illegal for the traveler to try to get travel insurance by providing incorrect, inconsistent, or incomplete information.

From the perspective of applicability, ethics could be classified into *macroethics* (large-scale, generic, coarser, and applies to organizations) and *microethics* (small-scale, specific, granular, and applies to individuals) [6]. The two are complementary, usually coexist, and are necessary for the practice of software engineering. It is possible to have one without the other. For example, the management of a software development company may engage in so-called 'ethics washing', but its requirements engineers may still act ethically.

B. Previous Work on Ethics in Software Requirements Engineering

The interest in integrating ethics in software RE and related areas is relatively recent and, in part, motivation for this paper. In [7], scenarios of unethical practices and their negative impacts on the users are given, and in [8] the need for requirements to reflect "socially responsibility" is underscored. To increase awareness of the ethical implications of software from a RE perspective [9], a systematic literature review and grey literature review was conducted in [10], and as number of ethical issues are highlighted. The *ACM/IEEE Software Engineering Code of Ethics and Professional Practice* (SECEPP) lists generic principles, each of which is refined into specific clauses, related to the behavior of and decisions made by professional software engineers as well as students of the profession. Finally, the *IEEE Standard 7010* provides guidelines for *Ethically Aligned Design* (EAD).

III. An Overview of a Conceptual Framework for Ethicality in User Stories

A. Ethicality as a Meta-Quality Attribute

A *meta-quality attribute* is a quality attribute about quality attributes. In this sense, ethicality is an anthropomorphic, extrinsic, meta-quality attribute, aiming to mimic certain aspects of *sentience* considered much desirable among humans.

There can be *degrees* of ethicality as an extrinsic property. For example, the *severity* of ethicality is especially acute in mission-critical applications, such as those that are high-risk safety-, privacy-, or security-critical, as opposed to low-risk casual applications.

As per model-based *software quality engineering* [11, 12], it is acknowledged that the notion of ethicality in and of itself is at a rather high a level to be useful, and therefore it needs to be decomposed into multiple, low levels to be meaningfully practical. This decomposition lends a hierarchical structure, as shown in Fig. 3. The mapping between extrinsic and intrinsic properties, as well as between intrinsic properties and entities of knowledge, is many-to-many (as evident also by Table 1). The knowledge entities could, for example, include principles, guidelines, patterns, and metrics.

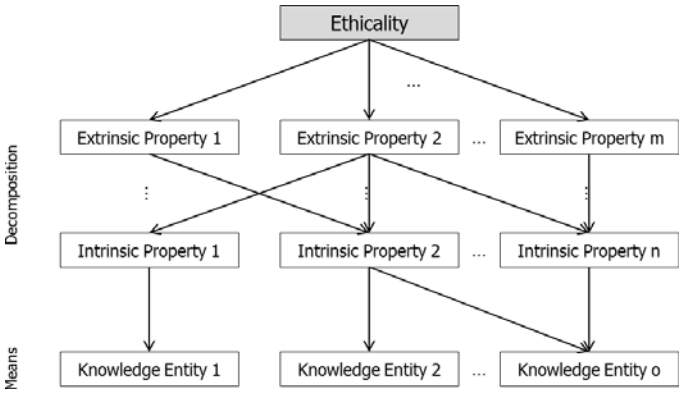

Figure 3. An abstract conceptual model for ethicality.

For example, *identifiability* (intrinsic quality attribute) contributes to *traceability* (extrinsic quality attribute) that, in turn, contributes to *transparency* (extrinsic quality attribute), and that, again, in turn, contributes to ethicality.

B. A Conceptual Meta-Model for User Story Environment

Fig. 4 shows a conceptual meta-model of a part of the user story environment from a managerial perspective. A user story is an aggregation of *role*, *goal*, and *value*, in that spatial order, and is associated with a *priority* and *acceptance criteria*. A user story together with its acceptance criteria is used to *estimate* the time and effort needed for completing it. The priority and estimate of a user story are based on the *risk* associated with that user story, which is of concern to both the users and the product owner [13, 14].

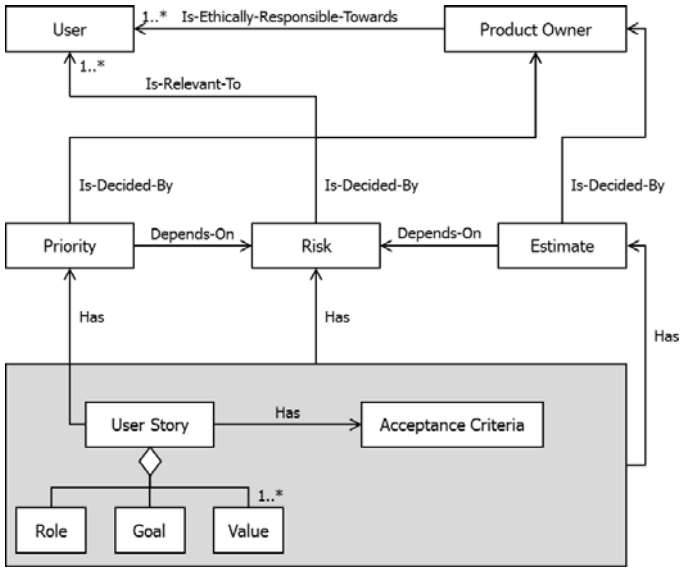

Figure 4. A conceptual meta-model of a part of the user story environment.

C. The Outline of an Ethically-Sensitive User Story Process

It is understood that certain desirable external quality attributes, such as safety, privacy, and security, cannot be addressed properly, if at all, at the end of development. It is therefore important that ethicality be considered at the beginning of a software development process, and remains an explicit concern throughout all stages of development.

Fig. 5 shows a conceptual meta-model for a minimal, continuous user story process [15], the activities and artifacts of which are aimed to be ethically-sensitive.

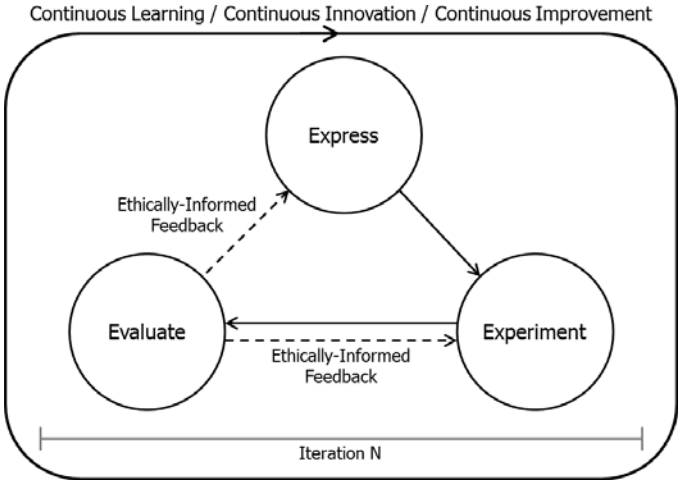

Figure 5. A conceptual meta-model for ethically-sensitive user story process.

The user story process proceeds as follows. The relevant stakeholders participate in *ethics poker*, following which the user stories are prioritized based on risk, and user stories that pose high risk with respect to ethicality are placed at high priority. The result of this step is a collection of *ethicality stories* for the current iteration. This is followed by the creation of a *value-centered* [16] and *responsibility-driven* [17] prototype for the ethicality stories by following the principles and patterns of *Design Thinking* [18] and *Systems Thinking* [19]. This experimental prototype could after a demonstration prompt a refinement of the user stories and possibly the elicitation of new user stories. Finally, there is an evaluation of the prototype involving actual users, for example, through crowdsourcing, observation, and/or survey, subject to *informed consent*. This could lead to feedback about ethicality of the product (which could prompt a refinement of the user stories and/or the prototype), as well as about the process (which could prompt an improvement of the user story process).

Table 1 lists a compendium of classical as well as novel quality attributes that are necessary for ethicality as identified by a number of recent studies [8-10, 20-22], and examples of corresponding user stories for a variety of application domains. It could be noted that the mapping between the set of quality attributes and the set of user story examples is many-to-many.

The challenges to the user story process include being able to elicit tacit or implicit knowledge from potential users, to mitigate cognitive biases (such as *Representativeness Bias*), and to be aware of ethical dilemmas (such as *Mission Impossible*) [23], and being able to control the accrual of *user story debt*, a type of *technical debt*.

TABLE I. EXAMPLES OF USER STORIES FOR ETHICALITY

Privacy, Security, Well-Being	• **US$_1$.** A visitor can access the cookie policy of a Web Site to make an informed decision about the data related to his or her visits. • **US$_2$.** A member can mark his or her profile as private to limit the information that can be shared.
Axiology, Fairness, Utility	• **US$_3$.** A customer can contact the administrator of the shopping system about the item return policy to be able to shop with surety and serenity. • **US$_4$.** A reader can distinguish between content and advertisement on the media portal to be able to discern accordingly.
Accessibility, Sustainability, Well-Being	• **US$_5$.** A patron with achromatopsia can navigate through the library system to be able to seek the books of his or her choice. • **US$_6$.** A student can report the presence of a malfunctioning critical user interface element on the course registration system so as to save time and effort of other students.
Responsibility, Transparency	• **US$_7$.** A project manager can check on the enterprise information system the daily calendar of all team members to be able to monitor their engagements.
Accountability, Explainability, Traceability, Transparency	• **US$_8$.** A maintainer can see on the source code management system the rationale associated with the status of (say, accepting or rejecting) a defect to be assured that the defect management process is being followed properly. • **US$_9$.** An auditor can independently access data used for performance benchmarks of a simulation system so that he or she can be assured of the correctness of the results and the consistency of the claims.
Competency, Traceability	• **US$_{10}$.** A programmer can independently access on the source code management system the review checklist against which his or her source code was reviewed so as to improve his or her programming capabilities.

D. A Survey on Software Engineering Ethics

To better understand the current state of knowledge and experience of software engineering ethics, including its relation to software quality, by those in academia and industry, a small-scale survey was conducted between Winter 2019 and Winter 2021. The respondees were from Canada, and consisted of graduate students in computer science or software engineering programs, and professionals in multiple software-intensive organizations, some of whom had been exposed to software engineering ethics. The survey had 16 items, each based on a 5-point Likert Scale ranging from Strongly Disagree (1) to Strongly Agree (5). Fig. 6 shows the results of 20 complete responses. It can be concluded from Fig. 6 that a majority of respondees understood the nocuous (I6) and innocuous (I11) impact of unethical behavior, and believed ethical behavior could change over time through guidance and training (I16), but there was no agreement on the responses on the issue of whether software engineering waste is unethical (I15).

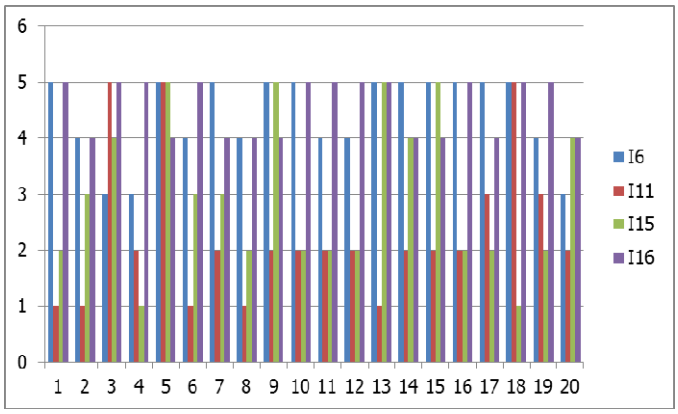

Figure 6. The distribution of selected responses from an ethics survey.

IV. Directions for Future Research

A. *"Who Dunnit?": Causal Analysis of Violations of Ethicality*

There can be a number of underlying causes of violations of ethicality: scarcity of resources, inadequate education in the application domain or ethicality engineering, inadequate elicitation of user needs, short-term expediency due to the pressure of time-to-market [24], lack of foresight, presence of (meta-)cognitive biases such as the *Dunning-Kruger Effect*, resorting to logical fallacies, gender inequity, politics, or deficiency of soft skills necessary for interviewing, negotiating, or reporting. Indeed, knowing the origins (or *root causes*) of such violations could be useful for a preventive approach towards ethicality, and is therefore of research interest.

B. *"Get 'em Early": Ethicality in Requirements Engineering Education*

The education that the students receive as learners of today creates, directs, and shapes their attitudes, habits, and temperaments as practitioners or researchers of tomorrow and beyond. For these traits to be socially-acceptable, ethics needs to be, as with the user story process, introduced *as early as possible* in the RE curriculum and emphasized throughout by lessons from history of software ethics, examples of ethical dilemmas, and case studies of ethical violations as per at least the SECEPP along with their potentially adverse consequences for society-at-large [25]. Indeed, a strong commitment to ethicality needs be a part of lifelong learning of all the students, and such "ethics literacy" needs to go beyond educational and professional contexts [26]. To investigate suitable approaches for doing so are therefore also of research interest.

V. Conclusion

The reasons for and the aspirations of software ethics are at least as relevant today as they were ~ 70 years ago. To be able to view ethicality as an extrinsic property of a software system lends itself to the established preventive as well as corrective knowledge in conceptual modeling, (agile) RE, and software quality engineering, as this paper has attempted to show.

The COVID-19 pandemic has led to a sobering reminder that a sole-effort of prevention or vaccination is insufficient, and that "it will take all of us". In a similar vein, ethicality may be realized to a notable extent only if it is perceived, discerned, and approached as a collectively-shared responsibility by all those who impact or are impacted by software. Having in place policies, processes, and procedures for ethicality is useful, perhaps even necessary, but these instruments have their inevitable limits [13, 26], even with the best of intentions and executions. In the end, the society may have to learn to live with the degrees of ethicality of software, of its own making.

Acknowledgment

This paper is dedicated to the indelible memories of Terrill Fancott and Peter Grogono, who passed away in 2020 and 2021, respectively. The authors are grateful to CUPFA for a Professional Development Grant, and to respondees of the survey for their interest and participation.

References

[1] T. W. Bynum, Computer Ethics: Its Birth and its Future. Ethics and Information Technology, 3(2): 109-111, 2001.

[2] C. Whitbeck, Ethics in Engineering Practice and Research, Second Edition. Cambridge University Press, 2011.

[3] T. Tamai, Social Impact of Information System Failures. Computer, 42(6): 58-65, 2009.

[4] D. Leffingwell, Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise. Addison-Wesley, 2011.

[5] A. E. Abbas, Next-Generation Ethics: Engineering a Better Society. Cambridge University Press, 2020.

[6] I. Nair and W. M. Bulleit, Pragmatism and Care in Engineering Ethics. Science and Engineering Ethics, 26: 65-87, 2020.

[7] B. Begier, Users' Involvement may Help Respect Social and Ethical Values and Improve Software Quality. Information System Frontiers, 12: 389-397, 2010.

[8] J. Cleland-Huang, Requirements That Reflect Social Responsibility. IEEE Software, 33(1): 109-111, 2016.

[9] F. B. Aydemir and F. Dalpiaz, A Roadmap for Ethics-Aware Software Engineering. The International Workshop on Software Fairness (FairWare 2018), Gothenburg, Sweden, May 29, 2018.

[10] A.-J. Aberkane, Exploring Ethics in Requirements Engineering. M.Sc. Thesis, Utrecht University, Utrecht, The Netherlands, 2018.

[11] W. Suryn, Software Quality Engineering: A Practitioner's Approach. John Wiley and Sons, 2014.

[12] S. Goericke, The Future of Software Quality Assurance. Springer Open, 2020.

[13] D. Hall, The Ethical Software Engineer. IEEE Software, 26(4): 9-10, 2009.

[14] P. Kamthan and N. Shahmir, A Characterization of Negative User Stories. The Twenty Eighth International Conference on Software Engineering and Knowledge Engineering (SEKE 2016), Redwood City, USA, July 1-3, 2016.

[15] P. Kamthan and N. Shahmir, A Framework for the Semiotic Quality of User Stories. The Twenty Seventh International Conference on Systems Engineering (ICSEng 2020), Las Vegas, USA, December 14-16, 2020.

[16] B. Friedman and D. G. Hendry, Value Sensitive Design: Shaping Technology with Moral Imagination. The MIT Press, 2019.

[17] R. Wirfs-Brock and A. McKean, Object Design: Roles, Responsibilities, and Collaborations. Addison-Wesley, 2003.

[18] H. Plattner, C. Meinel, and L. Leifer, Design Thinking: Understand – Improve – Apply. Springer-Verlag, 2011.

[19] J. E. Kasser, Systems Engineering: A Systemic and Systematic Methodology for Solving Complex Problems. CRC Press, 2020.

[20] V. Vakkuri, K.-K. Kemell, J. Kultanen, and P. Abrahamsson, The Current State of Industrial Practice in Artificial Intelligence Ethics. IEEE Software, 37(1): 50-57, 2020.

[21] R. Guizzardi, G. Amaral, G. Guizzardi, and J. Mylopoulos, Ethical Requirements for AI Systems. The Thirty Third Canadian Conference on Artificial Intelligence (Canadian AI 2020), Ottawa, Canada, May 13-15, 2020.

[22] M. D. Dubber, F. Pasquale, and S. Das, The Oxford Handbook of Ethics of AI. Oxford University Press, 2020.

[23] B. Berenbach and M. Broy, Professional and Ethical Dilemmas in Software Engineering. Computer, 42(1): 74-80, 2009.

[24] N. Patrignani and D. Whitehouse, Slow Tech and ICT: A Responsible, Sustainable and Ethical Approach. Palgrave Macmillan, 2018.

[25] J. Herkert, J. Borenstein, and K. Miller, The Boeing 737 MAX: Lessons for Engineering Ethics. Science and Engineering Ethics, 26: 2957-2974, 2020.

[26] K. Ryan, We Should Teach Our Students What Industry Doesn't Want. The Forty Second International Conference on Software Engineering (ICSE 2020), Seoul, South Korea, May 23-29, 2020.