Technical Debt is an Ethical Issue

J Paul Gibson[1][0000-0003-0474-0666], Massamaesso Narouwa[1],

Damian Gordon[2][0000-0002-3875-4065], Dympna O'Sullivan[2],
Jonathan.Turner[2], and Michael Collins[2][0000-0002-2034-2185]

[1] Telecom Sud Paris, Evry, France
[2] TUD, Dublin, Ireland
paul.gibson@telecom-sudparis.eu

Abstract. We introduce the problem of technical debt, with particular focus on critical infrastructure, and put forward our view that this is a digital ethics issue. We propose that the software engineering process must adapt its current notion of technical debt – focusing on technical costs – to include the potential cost to society if the technical debt is not addressed, and the cost of analysing, modelling and understanding this ethical debt. Finally, we provide an overview of the development of educational material – based on a collection of technical debt case studies - in order to teach about technical debt and its ethical implications.

Keywords: Technical debt, digital ethics, critical infrastructure, teaching

1 Introduction

Ward Cunningham first used the term technical debt to describe the trade-off between quality of code and time to deliver it (Cunningham, 2019). High-quality code costs more - in the short-term - to develop and deploy than low quality code. However, low-quality code costs more in the long-term due to higher maintenance costs. Thus, the delivery of low quality software incurs a debt that must be repaid later. From a purely pragmatic view-point, it is not always good business to make such repayments (Buschmann, 2011).We propose that technical debt should not just be considered a technological or business (Conroy, 2012) issue: it is an ethical issue.

The paper's main contributions are to raise awareness of the ethical issues that arise out of technical debt, and to report on a number of case studies that we carried out with relation to this issue. Our focus is on critical infrastructure, as the potential impact to society of not paying debt in such systems is very significant. However, we will also address technical debt whose impact is less general, but just as significant to individuals and members of specific groups. We also report on the development of an educational "brick" for teaching software engineers about ethics and how the software life-cycle can be improved through the integration of ethical requirements.

2 Technical Debt in Critical Infrastructure

Technical debt is most commonly used in reference to ICT systems (Tom et al., 2013), but the concept, and problem, is much older than that: eg, there is significant future cost associated with badly maintained infrastructure for critical systems such as transport, power, water, housing, health, education, etc. (Heimo & Holvitie, 2020). In our modern world, such infrastructure is dependent on underlying software systems (Rinaldi et al., 2001), which also incur significant debt if they are not maintained. Most of the research in this area has examined the risk associated with not keeping

such infrastructure secure, and more recently there has been discussion of the ethical issues with respect to security (Grady et al., 2021)

Although these aspects are important, a secure system is not guaranteed to function correctly. Poor quality software, and associated architecture is a major source of technical debt (Ernst et al., 2015) that requires specific tools and methods to be effectively managed (Avgeriou et al., 2020). Unfortunately, metrics for technical debt focus on the cost of maintaining the system, rather than the potential cost to society if the systems are not properly maintained. Just as the technical debt builds up interest over time, the potential risk to society may also grow dramatically; we refer to this as ethical debt.

Heimo and Holvitie have proposed an alternative definition of ethical debt to be the subset of technical debt associated with the direct cost of delaying properly analysing and understanding the ethical issues associated with the system currently under development (Heimo & Holvitie, 2020). A similar definition has been proposed when developing AI systems (Petrozzino, 2021). We shall widen this definition to include the potential cost to society if the technical debt is not paid.

3 Case Studies - Work in Progress

Our research objective is to investigate different ways in which the software development process could be improved by more rigorous modelling of ethical debt. As such, we are motivated and influenced by previous work on the social and human-centric aspects of software engineering (Tamburri et al., 2013, Spınola et al., 2019). We are following a case-study driven approach. There are a large number of technical debt case studies from which to choose, and our goal is not to do a literature review, such as seen in (Alfayez et al., 2020, Tom et al., 2013). Rather, we wish to re-evaluate some well-documented case studies from the point of view of ethical debt.

We have chosen not to consider the technical and ethical debt arising out of the use of AI (Bogner et al., 2021), but instead wish to focus on more traditional software systems. We wish to provide guidelines on improving the software process through better management of ethical debt, much in the way that different researchers have proposed better management of technical debt (Lenarduzzi, et al. 2021, Codabux & Williams, 2013).

3.1 Choice Of Case Studies

Following feedback from colleagues, and comments of the initial reviewers of the paper, we realised the importance of choosing the case studies that would give us most insight into the problem of the ethical issues arising out of technical debt.

First, we need to be clear what we mean by technical debt: it arisesfrom a deliberate choice to defer a technical cost in the short term, but which must be paid in the long-term. Implicit in this decision is that the technical debt will be managed and repaid. There are many different examples, with a range of consequences, eg:

1. Knowing that the system will not function correctly at some moment in the future, but waiting for nearer the time to fix it.
2. Knowing that the system does not function correctly for a small number of users, but it is deemed too costly to fix it just for them. Perhaps, in the future, they will allocate time and resources to fix this.

3. Knowing that the system architecture will not scale in the long-term, but waiting to fix this when the need arises
4. Knowing the code is insecure, but it functions correctly, and there is a decision to add new functionality rather than address the security issues
5. Knowing that code depends on packages/technologies which are out of the developer's control, but which they don't fully understand.
6. Knowing that the code has not been properly tested, but deploying it anyway with the plan to fix any issues that occur as they arise.
7. Knowing that the code is not well-documented, but delivering it even when it may be difficult to maintain.

We do not claim that all legacy systems have significant technical debt, but we do think that there is technical debt (of more or less importance) in all such systems. We agree with Monaghan and Bass (2020) that "*By positioning Legacy within the context of Technical Debt, practitioners have a more concrete understanding of the state of the systems they maintain*". Incurring technical debt is not necessarily unethical, but it may be in some cases. Furthermore, failing to properly manage the debt may also have important ethical consequences, and so incur significant dethical debt.

3.2 Mishandling of Dates – Y2K and beyond

The issue of dates is well-documented in the domain of software. The Y2K problem was widely reported (Williams et al., 2014) with worldwide concern over the possibility of critical bugs. Y2K was not the first such reported problem – issues with leap years were known much earlier (Neumann, 1992) – and it will not be the last, as we wait for Y2K38 (Okabe et al., 2020). Recently, more than 20 bugs have been reported with respect to Y2K22 (Neumann, 2022), including significant issues for Microsoft Exchange, Honda Clocks and Google Chrome users. Also, there has been some concern over the mishandling of leap seconds (Burnicki, 2015) in GPS systems (Anumasula et al., 2018).

We consider these to be examples of technical debt, as the issues were known at development time, and decisions were taken to wait until later to address them. The potential impact of not adequately addressing this debt before the strict deadlines is very significant, and could be considered critical. For example, the cost of fixing Y2K bugs is estimated to have been about 100 billion dollars worldwide; however, the cost to society of not fixing the bugs would have been much more (Best 2003). There are many reported examples of the consequences of failure to fix Y2K bugs on time, for example in critical health service code (Thimbleby, 2021). Thus, we consider this type of time-based technical debt to also be a serious ethical issue.

3.3 Open Source – Log4J

There is significant reliance on open source software, and difficulty in tracking the complex (inter-)dependencies (Bauer et al., 2020). The Log4J bug (Olbrich et al., 2020) is a good example of how a bug in a small, but significant, open source framework (used, in this case, for logging) can have far-reaching consequences (Srinivasa et al., 2022).

The technical debt, in this case is not specifically in the Log4J code, which was mostly written and maintained by a single person. The technical debt is in the

other systems that re-used the Log4J framework without fully analysing the dependency on the code and the risks of the code being insecure.

3.4 GDPR Compliance – Legacy Systems in The Health Domain

There are significant technical costs in "*Retrofitting GDPR Compliance onto Legacy Databases*" (Agarwal et al., 2021). Furthermore, verifying compliance with the different GDPR requirements is a significant challenge (Li et al., 2019). This may, in part, explain why GDPR non-compliance is a significant problem in healthcare (Agyei et al., 2020).

Would it be fair to blame this on technical debt ? This depends on whether GDPR was already 'on the horizon' when the legacy systems were being developed. If so, it could be said that there was a deliberate decision to incur technical debt with respect to future GDPR requirements. If not, then the question is not so clear cut. Perhaps the developers were aware of the security issues but, as there was no legal requirement to resolve them, they may have chosen to ignore them. In this case the future technical debt is an ethical issue. However, if the developers were unaware of such issues then this should not be considered to be technical debt.

There is evidence that companies prefer to pay fines for GDPR non-compliance than to pay off the technical cost of fixing noncompliance (Grant & Crowther, 2016). This is a clear ethical issue, particularly in critical domains such as health.

3.5 Accessibility

As with GDPR compliance, in many countries there are legal requirements for software with respect to accessibility. Unfortunately, these requirements are often not even met for public (web) services such as health (Alajarmeh, 2021), voting (Takahiro, et al., 2016) and education (Oswal & Hewett, 2013). Could this be another case of it being less costly to pay the fines than to pay the technical debt ? This merits further investigation, and is clearly an ethical (as well as legal) issue.

3.6 Agile Methods

There is some evidence that more agile development processes give rise to more technical debt (Behutive et al., 2017), and so this is an hypothesis that merits further, more rigorous, investigation. Agile development may also give rise to more ethical debt, with respect to the cost of performing the ethical analysis (Judy, 2009). Agile often leads to a 'Move Fast and Break Things' mentality, whose legal implications have been reported (Simon Chesterman, 2021). As software development is becoming even more agile - moving towards continuous integration and continuous deployment - it is clear that updating the software development process and life-cycle to include ethical aspects is critical.

3.7 Security

Significant security vulnerabilities can arise when security-critical code is deployed without having been properly tested. Three well-known examples of this are: OpenSSL's Heartbleed (Durumeric et al., 2014), Apple's "goto fail" (Bland, 2014),

and the INTEL AMT Vulnerability (Ermolov, M., & Goryachy, M., 2017). However, after brief analysis it was clear that this is not an example of technical debt as we define it: in none of these cases was there evidence of a deliberate decision taken to deploy code that was not properly tested. The problem was that the companies misbelieved that the deployed code had been rigorously tested. Thus, there was an issue with their software development process and incompetence, but this is not a good example of technical debt.

4 Future Work

4.1 Educational Brick(s) for Teaching about Technical and Ethical Debt

The development of the educational material will follow the approach taken in the EU Ethics4EU project (Gibson et al., 2021, Curley et al., 2021) which has focused on the pedagogic aspects of digital ethics, and the production of autonomous educational bricks. We believe that the chosen case studies will be highly motivational for the students, and provide open-ended problems for educators to leverage in teaching to a wide range of learners (Stavrakakis et al., 2021). We are currently developing project work based around the Log4J case study, and university web site accessibility issues.

4.2 Integrating Ethical Debt Analysis and Management in the Software Process

A long-term goal of our research is to propose methods for incorporating ethical analysis of technical debt into the software development process (at all stages in the lifecycle). As stated by Gotterbarn (1991), "*The ethical problems faced by the software engineer involve: the end product, the process of developing that product, and the human interactions in the development of the product.*"

We are also motivated by Biable et al. (2022), who propose that ethical issues should be included in the requirements phase of the software lifecycle. Previous research on management of technical debt during software development also suggests that the issue needs to be addressed throughout the life-cycle (Zengyang et al., 2015 and Rios et al., 2018)

4 Conclusions

We have shown, through a number of examples, that technical debt is often an ethical issue. There is an urgent need to educate engineers about technical and ethical debt, particularly with respect to critical system infrastructure.

Acknowledgements

The work reported in this paper was partially funded by the EU Erasmus+ transnational project Ethics4EU (http://ethics4eu.eu).

References

Agarwal, A., George, M., Jeyaraj, A., & Schwarzkopf, M. (2021). Retrofitting GDPR compliance onto legacy databases. Proceedings of the VLDB Endowment, 15(4), 958-970.

Agyei, E. E. Y. F., & Oinas-Kukkonen, H. (2020, April). GDPR and Systems for Health Behavior Change: A Systematic Review. In International Conference on Persuasive Technology (pp. 234-246). Springer, Cham.

Alajarmeh, N. (2021). Evaluating the accessibility of public health websites: an exploratory cross-country study. Universal access in the information society, 1-19.

Reem Alfayez, Wesam Alwehaibi, Robert Winn, Elaine Venson, and Barry Boehm. A systematic literature review of technical debt prioritization. In Proceedings of the 3rd International Conference on Technical Debt, pages 1–10, 2020. DOI = 10.1145/3387906.3388630

Anumasula, R., Mohanan, S., Rathour, H. K., Agarwal, P. K., & Baba, K. V. S. (2018, December). Indian Experience on Impact of Leap Second in Synchrophasor Systems. In *2018 20th National Power Systems Conference (NPSC)* (pp. 1-6). IEEE.

Paris C Avgeriou, Davide Taibi, Apostolos Ampatzoglou, Francesca Arcelli Fontana, Terese Besker, Alexandros Chatzigeorgiou, Valentina Lenarduzzi, Antonio Martini, Nasia Moschou, Ilaria Pigazzini, et al. An overview and comparison of technical debt measurement tools. IEEE Software, 2020. DOI = 10.1109/MS.2020.3024958

Andreas Bauer, Nikolay Harutyunyan, Dirk Riehle, and Georg-Daniel Schwarz. Challenges of tracking and documenting open source dependencies in products: A case study. Open Source Systems, 582:25, 2020. DOI = 10.1007/978-3-030-47240-5_3

Behutiye, W. N., Rodríguez, P., Oivo, M., & Tosun, A. (2017). Analyzing the concept of technical debt in the context of agile software development: A systematic literature review. Information and Software Technology, 82, 139-158.

Best, K. (2003). Revisiting the Y2K bug: language wars over networking the global order. Television & New Media, 4(3), 297-319.

Biable, S. E., Garcia, N. M., Midekso, D., & Pombo, N. (2022). Ethical Issues in Software Requirements Engineering. Software, 1(1), 31-52.

Mike Bland. Finding more than one worm in the apple. Communications of the ACM, 57(7):58–64, 2014. DOI = 10.1145/2622628.2622630

Justus Bogner, Roberto Verdecchia, and Ilias Gerostathopoulos. Characterizing technical debt and antipatterns in AI-based systems: A systematic mapping study. arXiv preprint arXiv:2103.09783, 2021. DOI = 10.1109/TechDebt52882.2021.00016

Burnicki, M. (2015, February). Technical aspects of leap second propagation and evaluation. In Requirements for UTC and Civil Timekeeping on Earth Colloquium. Science and Technology Series (Vol. 115).

Frank Buschmann. To pay or not to pay technical debt. IEEE software, 28(6):29–31, 2011. DOI = 10.1109/MS.2011.150

Simon Chesterman. 'Move Fast and Break Things': Law, Technology, and the Problem of Speed. Singapore Academy of Law Journal, 33:5–23, 2021. DOI = 10.1145/3244026

Zadia Codabux and Byron Williams. Managing technical debt: An industrial case study. In 2013 4th International Workshop on Managing Technical Debt (MTD), pages 8–15. IEEE, 2013. DOI = 10.1109/MTD.2013.6608672

Patrick Conroy. Technical debt: Where are the shareholders' interests? IEEE Software, 29(6):88–88, 2012. DOI = 10.1109/MS.2012.166

Curley, D. Gordon, I. Stavrakakis, A. Becevel, J.P. Gibson, and D. O'Sullivan. Adaptable and reusable educational bricks for teaching computer science ethics. In EDULEARN21 Proceedings, 13th International Conference on Education and New Learning Technologies, page 1991. IATED, 5-6 July, 2021 2021. DOI = 10.21125/edulearn.2021.0456

Ward Cunningham. The WyCash portfolio management system. ACM SIGPLAN OOPS Messenger, 4(2):29–30. DOI = 10.1145/157710.157715

Zakir Durumeric, Frank Li, James Kasten, Johanna Amann, Jethro Beekman, Mathias Payer, Nicolas Weaver, David Adrian, Vern Paxson, Michael Bailey, et al. The matter of Heartbleed. In Proceedings of the 2014 conference on internet measurement conference, pages 475–488, 2014. DOI = 10.1145/2663716.2663755

Ermolov, M., & Goryachy, M. (2017). How to hack a turned-off computer, or running unsigned code in intel management engine. Black Hat Europe.

Neil A Ernst, Stephany Bellomo, Ipek Ozkaya, Robert L Nord, and Ian Gorton. Measure it? manage it? ignore it? software practitioners and technical debt. In Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, pages 50–60, 2015. DOI = 10.1145/2786805.2786848

J Paul Gibson, Yael Jacob, Damian Gordon, and Dympna OSullivan. Developing an educational brick for digital ethics. In Moving technology ethics at the forefront of society, organisations and governments, ETHICOMP, pages 29–37. Universidad de La Rioja, 2021.

Gotterbarn, D. (1991, May). Ethical considerations in software engineering. In Proceedings of the 13th international conference on Software engineering (pp. 266-274).

Caitlin Grady, Sarah Rajtmajer, and Lauren Dennis. When smart systems fail: the ethics of cyber-physical critical infrastructure risk. IEEE Transactions on Technology and Society, 2021. DOI = 10.1109/TTS.2021.3058605

Olli I Heimo and Johannes Holvitie. Ethical debt in is development. comparing technical and ethical debt. ETHICOMP 2020, pages 29–30, 2020.

Jim Horning and Peter G Neumann. Risks of neglecting infrastructure. Communications of the ACM, 51(6):112– 112, 2008.

Ken H Judy. Agile principles and ethical conduct. In 2009 42nd Hawaii International Conference on System Sciences, pages 1–8. IEEE, 2009.

Valentina Lenarduzzi, Terese Besker, Davide Taibi, Antonio Martini, and Francesca Arcelli Fontana. A systematic literature review on technical debt prioritization: Strategies, processes, factors, and tools. Journal of Systems and Software, 171:110827, 2021.

Li, Zengyang, Paris Avgeriou, and Peng Liang. "A systematic mapping study on technical debt and its management." Journal of Systems and Software 101 (2015): 193-220.

Li, Ze Shi, Colin Werner, and Neil Ernst. "Continuous requirements: An example using GDPR." 2019 IEEE 27th International Requirements Engineering Conference Workshops (REW). IEEE, 2019.

Miura, Takahiro, et al. "Accessibility, efficacy, and improvements in voting methodology for visually impaired persons using a web-based electronic ballot system." Proceedings of the 8th Indian Conference on Human Computer Interaction. 2016.

Monaghan, B. D., & Bass, J. M. (2020, November). Redefining legacy: a technical debt perspective. In International Conference on Product-Focused Software Process Improvement (pp. 254-269). Springer, Cham.

Peter G Neumann. Leap-year problems. Communications of the ACM, 35(6):162–163, 1992. DOI = 10.1145/1349026.1349047

Neumann, P. G. (2022). Risks to the Public. ACM SIGSOFT Software Engineering Notes, 47(2), 4-7

Ryo Okabe, Jun Yabuki, and Masakatsu Toyama. Avoiding year 2038 problem on 32-bit linux by rewinding time on clock synchronization. In 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), volume 1, pages 1019–1022. IEEE, 2020. DOI = 10.1109/ETFA46521.2020.9212079

Steffen M Olbrich, Daniela S Cruzes, and Dag IK Sjøberg. Are all code smells harmful? a study of god classes and brain classes in the evolution of three open source systems. In 2010 IEEE International Conference on Software Maintenance, pages 1–10. IEEE, 2010. DOI = 10.1109/ICSM.2010.5609564

Catherine Petrozzino. Who pays for ethical debt in AI? AI and Ethics, pages 1–4, 2021. DOI = 10.1007/s43681-020-00030-3

Srinivasa, S., Pedersen, J. M., & Vasilomanolakis, E. (2022). Deceptive directories and "vulnerable" logs: a honeypot study of the LDAP and log4j attack landscape. In IEEE EuroS&P Workshop on Active Defense and Deception (AD&D). IEEE.

Steven M Rinaldi, James P Peerenboom, and Terrence K Kelly. Identifying, understanding, and analyzing critical infrastructure interdependencies. IEEE control systems magazine, 21(6):11–25, 2001. DOI = 10.1109/37.969131

Rios, N., de Mendonça Neto, M. G., & Spínola, R. O. (2018). A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners. Information and Software Technology, 102, 117-145.

Rodrigo O Spınola, Nico Zazworka, Antonio Vetro, Forrest Shull, and Carolyn Seaman. Understanding automated and human-based technical debt identification approaches-a two-phase study. Journal of the Brazilian Computer Society, 25(1):1–21, 2019. DOI = 10.1186/s13173-019-0087-5

Stavrakakis, I., Gordon, D., Tierney, B., Becevel, A., Murphy, E., Dodig-Crnkovic, G., ... & O'Sullivan, D. (2021). The teaching of computer ethics on computer science and related degree programmes. a European survey. International Journal of Ethics Education, 1-29.

Damian A Tamburri, Philippe Kruchten, Patricia Lago, and Hans van Vliet. What is social debt in software engineering? In 2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), pages 93–96. IEEE, 2013. DOI = 10.1109/CHASE.2013.6614739

Thimbleby, Harold. Fix IT: See and solve the problems of digital healthcare. Oxford University Press, 2021.

Edith Tom, Aybuke Aurum, and Richard Vidgen. An exploration of technical debt. Journal of Systems and Software, 86(6):1498–1516, 2013. DOI = 10.1016/j.jss.2012.12.052

S Mitchell Williams. An international investigation of associations between societal variables and the amount of disclosure on information technology and communication problems: The case of y2k. The International Journal of Accounting, 39(1):71–92, 2004. DOI = 10.1016/j.intacc.2003.12.003