When Cost-Benefits Go Wrong:
Differences in Decision-making between Physical Product and Software Defects

A Research Paper submitted to the Department of Engineering and Society

Presented to the Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science, School of Engineering

Navya Annapareddy
Spring, 2021

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Signature _____ Date _____
Navya Annapareddy

Approved __ _____ Date 5/4/2021_____
Dr. Hannah Star Rogers, Department of Engineering and Society

Abstract

The decision making process around physical products and non-physical and digital products like software can differ greatly, especially when considering decisions made around potential defects. The responses to a defect discovered during the technology or product development process can determine the success of that technology or product as well as the scope of the potential harm it causes. One main difference between nonphysical and physical products is that the consequences of nonphysical defects (such as an error in a software) are more difficult to relate to a concrete monetary or social loss than physical defects (such as manufacturing defects that are explicitly measured). This difficulty in abstraction is compounded by a lack of public understanding of technological mechanisms and components. An analysis of the differences in steps, scope, and systems of quantitative and qualitative-based decision making for defects for both physical and non-physical products is being conducted to highlight the differences in each process.

When Cost-Benefits Go Wrong:
Differences in Decision-making between Physical Product and Software Defects

Introduction

Successful responses to a defect discovered during the technology development process can entirely prevent negative consequences from occurring while inadequate responses can result in the total failure of a given technology (Shaw et. al, 2017). Actor Network Theory (ANT) is being applied in order to examine the actors, actions, and consequences of actions related to the elements of decision making in engineering. ANT is also being applied to identify the conflicting elements of the decision process and form an ethical hierarchy that encompasses and considers legal, economic, and professional perspectives and consequences of actions. The framework is also being used to explore the consequences of nonphysical defects (such as an error in a software) are different than physical defects (such as manufacturing defects that are explicitly measured). The technology decision-making process is complicated by both quantitative and qualitative dimensions, often in the form of metrics and social considerations. Using decision making tools that only consider quantitative factors leads to decisions that fail to consider externalities and the feasibility of attaching a valuation to elements of the decision making process that were inherently non-economic, such as human lives (Majumder & Madheswaran, 2017). On the other hand, adding social components to the majority of decisions is often seen as a hindering and slowing addition to the decision making process and thus the technology development process (Lyden et al., 2010). An analysis of the steps of quantitative and qualitative-based decision making will be conducted to highlight the differences in each process and points of integration for decision makers.There is a difficulty in abstraction for nonphysical is compounded by a lack of public understanding of technological mechanisms and components

(Yapo & Weiss, 2018). In order to integrate social dimensions into the engineering decision making process, the following question must be answered: what are the principles of social responsibility in engineering and how can ethical decision hierarchies in engineering be developed to contextualize decision making tools and steps?

Physical Product Development Cycle

Physical products are items that have physical components that are manufactured, assembled, or otherwise physically created. In a technical context, this would include computer hardware, display screens, and sensors. These products are often meticulously constructed with quality control standards for the final product as well as manufacturing conditions (ie: strict guidance on ensuring there is no dust in the air). For other products such as clothing or artwork, much of the manufacturing or assembly is still done by human workers rather than automated machines or in factories.

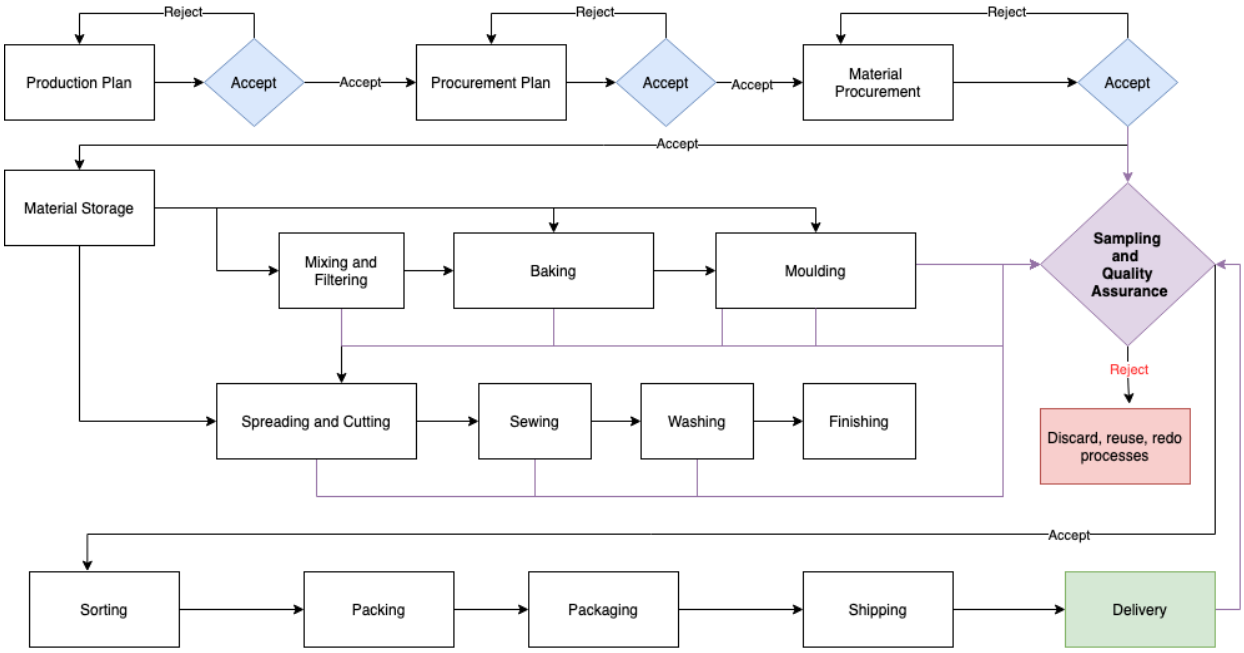

Figure 1. Decision Action Diagram of a typical product development cycle for physical products.

Many products do not solely rely on one method of manufacturing and instead each step of the total manufacturing process requires a different method of production. Regardless of the actual manufacturing process, a typical product development process for physical products begins well before actual production through plans for material procurement and production. As depicted in Figure 1, a typical physical product development process encompasses many interdependent stages of production that terminate after successful exporting of the product. During production, it is customary to sample products for quality control and adjust the manufacturing process. After product delivery, quality control can still be conducted by way of consumer reporting or regulatory testing.

Software Development Cycle

The software development life cycle (SDLC) is a heavily iterative process. Rather than specific product development steps, it consists of more open ended stages of development such as prototyping and software development. Software dependencies, environments, and requirements frequently change so development is highly agile. This manifests itself in the testing processes of the SDLC as well; both large scale tests and small scale tests based on a single development task occur. Noticeably, quality assurance is typically regarded as an internal regulation task rather than an external one. The metrics a given development team may pursue varies, as do the specific measures of error tolerance.

As depicted in Figure 2, unlike physical product development, software development cycles rarely truly terminate. Instead, software reaches a phase of maturity where maintenance rather than primary development is pursued.

While physical and nonphysical products are developed separately from each other, it is very rare that no physical products (ie: computer hardware) are present during the SDLC or that

no software products (ie: manufacturing control software, payroll software) are present during physical manufacturing. Despite this, each product or subcomponent is able to be considered individually in the quality control process.

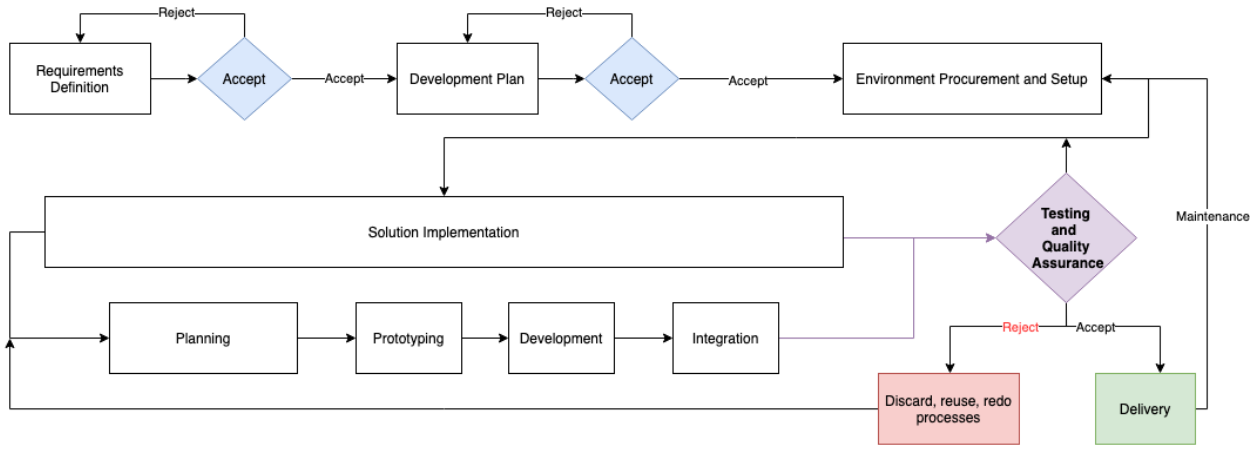

Figure 2. Decision Action Diagram of a typical software development cycle

The implications of an integrated process containing both physical and software features include the fact that defects in one type of product can interfere or amplify the defects in another. For example, computer vision models are common in the manufacturing industry to identify and even predict defects while product components are being transported or modified. These models are nonphysical software but a biased or inaccurate model could cause physical defects in the product being manufactured to go unnoticed, thereby increasing the total number of physical defects. This is a cause for concern as software and machine learning models take on increasingly regulatory and control purposes.

Similarities in Product Development Cycles

An ANT approach detailing the actors of the decision making process of each type of product development is also helpful in understanding the similarities and differences of defect identification and response. In a given network, subjects (not all with equal roles or power)

interact with objects and mediating artifacts under a larger community, guiding set of rules, and a division (even if it is unequal) of labour (Jørgensen, 2017).

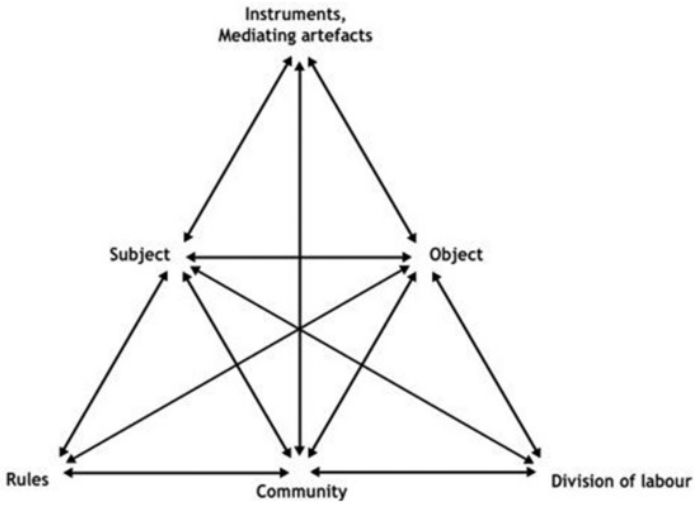

Figure 3. Actor-Network representation of complex activity systems. Reprinted from "Reframing tourism distribution - Activity Theory and Actor-Network Theory," by M. Jørgenson et. al, 2017, *Tourism Management*, October, 2017. Copyright 2017 by Elsevier Ltd. Reprinted with permission.

While physical product and software processes differ, there remain similarities in their development cycles. At an organizational level, many engineering and manufacturing companies or groups' cultures (communities in the actor network) are results and metrics driven (Wilson, 2020). This poses a competing value system to one centered around social responsibility, meaning that the corresponding rules and division of labour are conducted according to the value system of the most powerful actor(s) or communities. Adding social components to the majority of decisions is often seen as a hindering and slowing addition to the decision making process and thus the technology development process regardless of product type (Sweeting, 2018). This can lead to steep requirements and limits on how much money or how many resources can be allocated for quality assurance or defect identification. In the case of the Pinto, Ford (an example of a community actor) set strict limits of cost (rules for engineers) for production of the Pinto (no

more of $2000 per car) which limited the production team (subjects in the network) to follow up on issues found during safety testing (Leggett & Palmiter, 1999).

Additionally, both types of product development cycles indirectly and directly interact with external actors. Regardless if their work concerns software or physical products, engineers and trade professions are often bound to codes of ethics through memberships of professional and academic groups. Such codes include the Code of Ethics maintained by the National Society of Professional Engineers (NSPE) and similar organizations (National Society of Professional Engineers, 2019). These ethical guidelines might range in scope from explicitly prohibitions and zero-tolerance policies (ie: plagiarism) to implicit and more vague rules of practice (ie: NSPE's mandate for engineers to act for each employer as faithful agents or trustees). In many cases, these codes come into conflict with each other or the law and legality of actions such as whistleblowing. Generally, these codes do not specify between physical and nonphysical defects. Instead the ethical stipulations are related to scope of work any given member or actor is involved in, which frequently include both physical and nonphysical products.

Differences in Product Development Cycles

The ANT method supports answering the research question by examining the actors, actions, and consequences of actions related to the elements of decision making in engineering. The method focuses on the relationships between actors (not-necessarily mutually human) and will allow the conflicting or differing elements of the decision processes to be addressed. In doing so, it will help lead to an ethical hierarchy being formed that encompasses and considers many different perspectives and consequences of actions.

While both physical product and software development involve interdependent and even recursive steps, they highly differ in the order and scope of processes as well as their

corresponding actor networks. Software development is by nature both an adaptive and predictive process. When new environments, tools, and dependencies become available or updated, software must be updated or modified by engineers (human actors) accordingly to avoid becoming obsolete or nonfunctional. Processes in physical product development can also demand change but it is unlikely that large scale changes such as rehauling equipment or automotive processes will occur frequently if at all. Instead, physical product development focuses on quality assurance at each step of development with stringently documented error tolerances for widely used and accepted metrics, such as defective product counts or physical attributes of a part (ie: radius). This assurance process can be done by human actors or as an automated process via software or machine features. Conversely, other techniques solely focus on the behaviors of human actors to reduce defects by controlling their response to minute changes in control metrics rather than large scale implementation change. For example, Six Sigma is a frequently employed business and manufacturing strategy that emphasizes human awareness of the manufacturing practices and encourages human actors to measure and act on deviations from the ideal using statistical process control and avoiding workplace designs that cause attention deficits or instances of human error. In this case, Six Sigma can be considered a mediating process that sets the rules of subjects in the actor network of manufacturing.

Similarities in Decision Making Processes Around Defects

Regardless of product physicality, engineering decision making processes utilize many types of decision tools and many are dependent on the type of product being developed. Information control methods focus on the actual gathering of data (ie: user surveys) while paradigm models are models of thought (ie: Strengths, Weaknesses, Opportunities, Threats SWOT analysis). These models are usually established frameworks that are edited for brevity

and superior recall for widespread use (ie: The 4 Ps of Marketing). When a defect arises, simulation methods such as a cost-benefit or critical path analysis are often used. Methods are also used to weigh different decisions, such as conflict analysis and performance matrices. These decision contrast tools often involve assigning weights to the severity of certain consequences and ranking how well a decision avoids or minimizes negative effects. When managing defects, quality management tools such as histograms and control charts where limits are set for the number of acceptable defects are employed. For physical manufacturing processes, this would entail physical defects in products such as cosmetic defects or diminished functionality. For software, defects would be instead related to metrics such as system downtime, efficiency, or inaccurate predictions or results. Not all decision making tools lack a social component, but some methods like the cost-benefit analysis are prone to requiring subjective monetary assignments to actions if a component is non-monetary in nature. Not all decision making tools are supervised by human actors or include social considerations at every step. This could result in misleading analysis results or results that ignore the indirect and social costs of elements in the analysis (Institute for Manufacturing, 2016).

Product defects also have legal implications as legally, social responsibility around decision making is liability and negligence focused. The principle of comparative negligence in law states that if an accident occurs, every party or actor involved is liable for their contributions to the accident (Chung, 1993). When decision making methods are utilized when considering engineering defects, decision makers often focus on what decision, if made, will make them least vulnerable to litigation and financial settlements. Defendants of a case are also subject to the Balancing approach, which states that a defendant is not liable for not taking precautionary measures for incidents that have a small chance of occuring and a cost related to preventing it.

Financially, decision making tools are utilized in weighing the economic elements of a decision. A decision made around a defect will either result in a net loss of money, a net gain of money, or no change. To determine a desired quantitative decision threshold, every element included in the cost analysis must relate to a quantitative metric, usually cost. In the cost-benefit analysis carried out by Ford (a community actor) when making decisions around the Pinto (an object in the actor network) design's fuel tank defect, the financial elements of the decision process included the cost to implement the safer design per car (about $11), the number of vehicles the design would apply to (about 13 million vehicles), and the financial benefit of doing so. The cost of implementing the design over the number of vehicles it applied to totalled to about $137 million. To calculate the financial benefit implementing the design, Ford chose to use the financial impact of the lives predicted to be saved. In order to compare benefits and costs of the decision to be made, Ford then used figures given by the National Highway Traffic Safety Administration to quantify the cost of a death due to the old design as $200,000, the cost of an injury as $67,000, and the cost of vehicular damage as $700 per vehicle (Leggett & Palmiter, 1999). The calculated benefits to society totalled about $50 million while the costs to Ford were $137 million, therefore, leading Ford to conclude they were justified in not implementing the safer design. By using decision making tools that consider only one type of factor (quantitative) and require all terms to be expressed in a common metric (dollars), Ford failed to consider externalities and the feasibility of attaching a valuation to elements of the decision making process that were inherently non-economic. Such externalities include the negative publicity Ford received by external actors like the public and government due to the accident litigation of the Ford Pinto. Cost estimation tools are also used in the SDLC albeit in a different context. Rather than physical product changes being estimated, SDLC costs and benefits would more

likely involve personnel and hardware or development environment costs. The legal and behavioral incentives to make decisions about product defects on the basis of metrics discourages elements of social responsibility to be included in decision making methods regardless of if a given defect belongs to a physical or software product.

Differences in Decision Making Processes around Defects

While both physical and software product development tools utilize decision making tools, there exists a key difference in their use. Physical manufacturing typically relies on scale and volume of production. Thus, when a process change is being considered it's individual cost is amplified several times (even millions of times) as in the case of the Ford Pinto. Software development on the other hand is a front loaded effort, not a scale-dependent procedure. While larger systems require more effort to maintain and initially parallelize, they eventually mature to the point of maintenance and are marginally more time consuming or difficult to manage per volume increase. This means that in a given time period, for physical products both the costs and benefits of a process change increase but for software the costs do not increase (or only marginally increase) once primary development is over while the benefits remain not only for existing systems but also for future ones.

For physical processes, the division of labour in the actor network is segmented and mediating artifacts such as control displays and statistical software are often responsible for large scale defect identification while human subjects (actors) are responsible for sampling, documentation, and enforcing process control. For the SDLC, division of labour in the actor network is less stratified and defect resolution can not only be an ongoing process but an unregulated one. This is because SDLC teams tend to have a limited number of human actors

(subjects) interacting with the product (object) with a limited number of rules (especially regarding data security and intellectual property protection) and mediating artifacts.

For physical manufacturing, control metrics are continuously collected, reported, and acted on by human actors. If a given batch or sample of a product does not meet specifications by a large amount, the group of products that it belongs can be scrutinized and even discarded. Conversely, during the SDLC evaluation metrics can only be collected following successful implementation if at all. Prototyping a development solution can take up the majority of the SDLC process and analysis and evaluation is often the focus of the subsequent maintenance phase rather than the primary development phase. This can be an issue because an architecture of an implemented solution may be conducive or prone to errors but cannot be easily changed once deployed. Software evaluation metrics, like the SDLC itself, are often internally managed. If a specific defect were to occur, it is more difficult to narrow down a specific root cause in a given software than it is to reference a specific feature that failed in a physical product given the nature of the actor network and lack of mediating instruments. Software itself is a system of interdependent features that rely on other software and physical systems (such as backup data servers and accurate computer or sensory hardware). Because software evaluation metric collection is a non-explicitly regulated process, it is also possible for a developer or organization to claim that they were unaware of a given defect or its scope. Realizing this, many software developers require users and customers alike to accept their own terms and conditions before utilizing any implementation of their work.

The inclusion of social responsibility in decision making around engineering defects is complicated through its open ended nature. While both physical and nonphysical defects vary in magnitude, nonphysical defects (such as the bias in a machine learning model) are more abstract

(Suresh & Guttag, 2019). Defects range from logical defects (such as arithmetic errors in a software) to large scale performance issues (such as a machine learning model that does not create the expected output, or creates incorrect output). Physical products for the most part adhere to control practices that are easily related to monetary losses or issues of negligence. The same set of techniques and strategies that have been implemented for manufacturing and physical product development cannot be used to the same effect for nonphysical products like software. While manufacturers interact regularly with regulatory agencies and even each other to develop and even publicly make available control metrics, software development is a largely internally regulated process fraught with intellectual property concerns that apply not only to the product but also any metrics its SDLC utilizes. This creates an informational gap between defect response in physical and software development as the less there is a widely known and accepted standard for acceptable error tolerance, the more such metrics are subjectively measured, evaluated, and responded to.

Conclusion

By considering ANT, it is clear that physical product development contains a more established and hegemonic means of quality assurance while software development is a vastly more organization-dependent and internal process. While software development changes are front-loaded (meaning the cost of changes are marginal after primary development), physical product process changes create an increase in both costs and benefits over time (meaning the cost of changes grow according to the scale of production). This means that physical product developers might be more resistant to process changes since they will be expensive to scale while software developers might be resistant to process changes due to the non-mandatory and subjective nature of error tolerance in the software life cycle. Consequently, even though the

same decision approaches or tools are applied to a given process, the decision made is influenced by the type of product.

Finally, from a legal and ethical perspective, software defects are present at a higher level of abstraction and are thus harder to consider as the cause of concrete harm (monetary, physical, or otherwise). In terms of ethical implications, it would follow that determining or proving negligence for software is more difficult than for physical products. This difficulty is extended into legal stances on defending or prosecuting those responsible product defects as successful arguments hinge on the presence (or lack) of evidence and clear causal circumstances. This would indicate it might be beneficial for regulating bodies similar to those present for physical product development to be maintained although this has implications on the balance between regulatory infringement and an organization's power. This analysis would support a more clear definition of negligence for software and machine learning as the existing concepts fail to address ambiguity in how negligence can be considered for nonphysical products. Ultimately, it can be argued that physical products and software products have distinct developmental processes. While these processes do sometimes share elements (such as competitive organization culture and protection of intellectual property), they differ starkly in their approach to evaluation metrics and quality assurance which manifests itself into distinct methods of responding to and resolving defects. Both decision making processes will continue to evolve as more integrated systems of both physical products and software emerge and become increasingly interdependent and complex. In order to maintain an appropriate level of process integrity, it will be critical for regulators and product developers alike to distinguish between the defect responses of physical and digital products.

References

Baiocchi, G., Graizbord, D., & Rodríguez-Muñiz, M. (2013). Actor-Network Theory and the ethnographic imagination: An exercise in translation. *Qualitative Sociology, 36*(4), 323-341. doi:10.1007/s11133-013-9261-9

Chung, T. (1993). Efficiency of Comparative Negligence: A Game Theoretic Analysis. *The Journal of Legal Studies, 22*(2), 395-404. Retrieved October 15, 2020, from http://www.jstor.org/stable/3085587

Hwang, S., Park, J., Kim, N., Choi, Y., Kweon, I.S., 2015. Multispectral pedestrian detection: Benchmark dataset and baseline, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition

Institute for Manufacturing. (2016). *Technology Management Decision Support Tools*. Institute for Manufacturing Technology Policy. https://www.ifm.eng.cam.ac.uk/research/dstools/.

Kirkman, R. (2004, August 1). The Ethics of Metropolitan Growth: a Framework. Philosophy & Geography, 7(2), 201 - 218.

Jørgensen, M. T. (2017). Reframing tourism distribution - Activity theory AND Actor-Network Theory. *Tourism Management, 62*, 312-321. doi:10.1016/j.tourman.2017.05.007

Leggett, C., & Palmiter, A. (1999). THE FORD PINTO CASE: THE NEGLIGENCE-EFFICIENCY ARGUMENT. Retrieved from https://users.wfu.edu/palmitar/Law&Valuation/Papers/1999/Leggett-pinto.html

Livieris, Ioannis E., Emmanuel Pintelas, and Panagiotis Pintelas. "A CNN–LSTM Model for Gold Price Time-Series Forecasting." *Neural Computing and Applications*, April 13, 2020. https://doi.org/10.1007/s00521-020-04867-x.

Lyden, P. D., Meyer, B. C., Hemmen, T. M., & Rapp, K. S. (2010). An ethical hierarchy for decision making during medical emergencies. *Annals of neurology*, *67*(4), 434–440. https://doi.org/10.1002/ana.21997

Majumder, A., & Madheswaran, S. (2017). Meta-analysis of Value of Statistical Life Estimates. IIM Kozhikode Society & Management Review, 6(1), 110–120. https://doi.org/10.1177/2277975216678546

National Society of Professional Engineers. (2019). *Code of Ethics Examination*. https://www.nspe.org/resources/ethics/ethics-resources/code-ethics-examination.

Reed, M. (2018). The Classification of Artificial Intelligence as" Social Actors" (Master's Thesis). Georgia State University, GA.

Shaw, W. H., Barry, V. E., Issa, T., Catley, B., & Muntean, D. (2017). *Moral issues in business*. Cengage Learning.

Suresh, H., & Guttag, J. V. (2019). A framework for understanding unintended consequences of machine learning. *arXiv preprint arXiv:1901.10002*.

Sweeting, B. (2018). Wicked problems in design and ethics. In P. H. Jones & K. Kijima (Eds.), Systemic design: Theory, methods, and practice (pp. 119-143). Tokyo: Springer Japan. DOI: 10.1007/978-4-431-55639-8

Wilson, T. (2020). Instagram, Amazon, and Machine Learning: Ethical Implications of Collecting and Analyzing Commercial User Data (Unpublished undergraduate thesis). University of Virginia.

Yapo, A., & Weiss, J. (2018). Ethical Implications of Bias in Machine Learning. Hawaii International Conference on System Sciences 2018 (HICSS-51). Retrieved from https://aisel.aisnet.org/hicss-51/os/topics_in_os/