Dark Patterns for Unethical Software Engineering

Cesare Pautasso
c.pautasso@ieee.org
Software Institute, USI, Lugano, Switzerland

Abstract

Unethical software engineers write software to satisfy harmful requirements. While patterns promote beneficial solutions to recurring problems, dark patterns intentionally introduce harmful solutions. In this paper we present a collection of 16 dark patterns widely used by unethical software engineers to violate users privacy (email pixel injector, stealthy input logger), pursue monetization at all costs (aggressive advertiser, ad-blocker detector, pay to win, artificial scarcity hoarder, DRM rug puller, obsolescence planner), commit digital frauds (cybersquatter, sneaky terms degrader, interoperability breaker), manipulate search rankings (fake review generator, search ranking kickbacker), and engage in unethical artificial intelligence practices (training data harvester, bot pretender, deceptive deepfaker). By discussing the ethical consequences of each pattern we aim to raise awareness about them and encourage their avoidance by ethical software engineers, architects and practitioners.

CCS Concepts

• Social and professional topics → Codes of ethics; **• Software and its engineering → Design patterns**.

Keywords

Ethical Software Engineering, Dark Patterns

ACM Reference Format:
Cesare Pautasso. 2024. Dark Patterns for Unethical Software Engineering. In *Proceedings of 31st Conference on Pattern Languages of Programs, People, and Practices (PLoP 2024).* ACM, New York, NY, USA, 14 pages.

1 Introduction

While the benefits of digitalization [23, 70] have been widely studied in terms of improved efficiency, freedom of expression [56], knowledge sharing at scale, education opportunities, and democratic participation [97], information technology and software engineering also inherently lead to some disruptive and deleterious consequences [100]: echo chambers [68], infant addiction [51], biased and opaque algorithmic decision making [71], surveillance [108], monetization at all costs, and novel kinds of fraud enabled by emerging technologies such as blockchain and artificial intelligence [19, 59].

In this paper we collect a small set of 16 dark[1] patterns [63] which highlight how some of these problematic aspects are obtained by intentionally introducing the patterns in concrete software systems and applications. The patterns have been too often applied in practice by unethical software engineers who do not value autonomy, fairness, inclusiveness, privacy, respect, trust, transparency, and sustainability. Raising awareness about them can hopefully start a discussion on how to best avoid them in the future as the ethics, trustworthiness and responsibility of software engineers become even more critical [27, 45, 83].

Raising awareness about them can also unfortunately lead to a *dual-use dilemma*, in which the presence of these dark patterns is intensified and promoted, also because of how effectively they work exactly as intended. While software engineers are trained and expected to conceive, design, develop, build, test and release high quality software which satisfies its functional requirements, who bears the full responsibility when the software implements harmful requirements? "Engineers should serve neither as servants, blindly taking orders from management or society, nor as overlords who paternalistically know best what society needs. The idea of 'the Great Engineer' went out with Herbert Hoover. The idea of the engineer as order-taker went out with Nuremberg" [86, 87]. As we write this collection of dark patterns, like researchers on security patterns [88] before us, our intention is far from leading software engineers (or students entering the field) into temptation. On the contrary, this collection is meant 1) to raise a warning flag; 2) to stimulate a discussion on where exactly to draw the line; 3) and hopefully lead to their avoidance, or at least some form of resistance against the identified malicious practices.

Each dark pattern is positioned in its context, described with the usual problem and solution, followed by a discussion of the consequences of introducing it and an explicit argument summarizing the ethical violation. A few references to generic known usage scenarios complete the pattern template. While specific known uses are available upon request, we chose not include them since they are probably well known already and their selection would be arbitrary: it is outside the scope of this paper to apportion blame. We leave it as an exercise to the reader to spot real world examples of these dark patterns in the wild.

It is also outside of the scope of this paper to determine whether some of the described unethical practices are illegal or not, as this both depends on the current legislation and the regions in which it is in force.

The pattern collection is organized in five categories (Table 1): covert surveillance, monetization at all costs, digital fraud, ranking manipulation, and unethical artificial intelligence practices.

[1]We chose this term as opposed to "anti-pattern" in agreement with [102]: while both should be avoided, dark patterns are intentionally introduced to cause harm, while anti-patterns can be seen as design mistakes which can be corrected through a refactoring.

Table 1: Overview of the 16 Dark Patterns for Unethical Software Engineering

Dark Pattern	Problem	Solution	Ethical Violation
		Covert Surveillance	
1. Email pixel injector	How to covertly track whether email messages have been opened for reading by their recipients?	Embed tracking pixel-sized image in HTML-formatted emails	Privacy violation
2. Stealthy input logger	How to explain the behavior of users without asking them?	Covertly capture partial input even before it is submitted for processing	Privacy violation; exploit users as testers
		Monetization at all costs	
3. Aggressive Advertiser	How to maximize ad revenue?	Display ads everywhere	Profits over user experience
4. AD-blocker detector	How to force blocked ads to be displayed?	Refuse to display content when ad-blocker is detected	Users forcefully exposed to ads
5. Pay to Win	How to make gamers pay to play games?	Give away games for free, but make it impossible to win without paying	Gamers turn into gamblers
6. Artificial scarcity hoarder	How to intentionally restrict the availability of digital goods to manipulate their price and increase profits?	Make digital goods behave like physical ones	Exploit fear of missing out
7. DRM rug puller	How to make users pay again for DRM-protected digital goods already bought?	Stop the digital rights management (DRM) authorization system	What was deceptively “bought” vanishes into thin air.
8. Obsolescence planner	How to keep profiting from a saturated market?	Create a need for unnecessary upgrade	Waste limited natural resources
		Digital Fraud	
9. Cybersquatter	How to take advantage of users who trust the mistaken URLs they open?	Register mis-typed domain names	Exploit users trust
10. Sneaky terms degrader	How to comply with the requirement to inform users about changes while making it unlikely for them to withdraw if they no longer agree?	Inform users that a new agreement will be introduced without specifying in detail what changed	Mislead users into accepting new, possibly harmful terms
11. interoperability breaker	How to ensure full control over the client application user experience by the API service provider?	Block, intentionally break or simply stop responding to unwanted clients	Waste client developers time and effort
		Ranking Manipulation	
12. Fake review generator	How to influence the aggregate review scores?	Automatically post a large number of positive or negative reviews	Mislead users; ruin recommender’s reputation
13. Search ranking kickbacker	How to ensure arbitrary content is ranked in the top search results?	Obtain a high ranking by paying a fee	Search engine integrity harmed when organic and sponsored results look the same
		Unethical AI Practices	
14. Training data harvester	How to amass a sufficiently large quantity of training data suitable for a variety of applications?	Crawl every possible data source, even without authorization	Exploit original authors of training data
15. Bot pretender	How to transparently deal with limitations or failures of artificial intelligence bots and avoid accountability?	When the bot fails, have a human silently take over	Make the bot appear smarter than it really is; skirt responsibility for the human taking over
16. Deceptive deepfaker	How to deceive viewers with misinformation which appears to be authentic?	Develop software applications that produce deep-faked videos and post them as authentic	Take over someone’s voice or image without permission

2 Covert Surveillance

Surveillance was originally practiced by governments to watch over their subjects [60]. Existing practical limitations on the targeted population size and the breadth of the surveillance scope have been swept away by digital technology [53]. Now surveillance tools are widely deployed and provide the foundation of large sectors of the economy [108].

In this section we include two dark patterns where covert surveillance is used to track email communication and to extract feedback from users. In both cases users are not aware they are being watched as they read "their" email or use "their" software.

Dark Pattern 1 - Email Pixel Injector

Context:

Email communication does not provide guaranteed message delivery [72]. While specific receipt notification delivery methods exist, users may choose to disable them. Massive marketing lists from senders of unsolicited commercial email – spammers – have a pressing need to precisely track which messages have been delivered to whom as part of different campaigns.

Problem: *How to covertly track whether email messages have been opened for reading by their recipients?*

Solution:

Send messages using only HTML format (not in plain text) so that it is possible to embed within the document tiny invisible images. These pixel-sized images will be fetched from a dedicated message delivery tracking server. This server will be pinged by the user email client at the time (and in some cases, every time) the recipient opens the message for reading it.

Consequences:

It becomes possible to distinguish who does not read a given email message from who actually opens the message for reading it. The sender can know not only whether the message was delivered or not, but also when exactly the user opened it for reading it, and use this knowledge for timing the delivery of follow up messages. Senders may for example send a creepy complaint to recipients that opened their message but – for some reason – did not (yet) reply to it. By looking up the geographic location of the user IP address, it becomes possible also to estimate the physical location of the recipient at the time the email was opened for reading.

Ethical Violation:

The privacy of the email recipient is compromised. Email users are no longer free to check and read their email at their own pace. Even email users that reject sending receipt notifications are still tracked as they display the email content.

Drawing the line:

There are standard email protocols which allow senders to be notified of the receipt of their message, if recipients allow it [34]. If they disallow notifications to be sent back, there is probably a good reason. For example, they do not wish for their actions to be monitored, their mail reading habits to be inferred. They do not wish for the recipient to know that they have already seen the message, even if they are not going to reply to it yet.

Injecting tracking pixels into mail messages crosses the line as it intentionally tricks the mail reader application into sending out unwanted notifications even if the user may have disabled this behavior.

Known Uses:

Every professional mailing list management software offers delivery tracking features based on spy pixels [46]. Some email clients offer protection by not displaying images by default (but if users choose to display images, they will do so indiscriminately). Others will do so by proxying the image request through the email provider, thus hiding the IP address of the reader. A few have recently started to detect and remove such spy pixel images from known trackers and to warn users about their presence.

Dark Pattern 2 - Stealthy input logger

Context:

Users may perform only the initial steps of a workflow without reaching its natural completion (e.g., search for products, add them to a shopping cart, but stop before submitting the order [96]). Users may begin to enter information into a form but may forget to submit it, or intentionally stop before doing so. Users are not willing or able to provide explicit feedback about their actions.

Problem: *How to explain the behavior of users who leave tasks incomplete without asking them?*

Solution:

Covertly capture partial user input so that behavior of explicitly identified users can be observed before they reach their task completion. If necessary, log every click [66], keystroke [15], and sample any available device sensor [18] during the task execution.

Consequences:

Obtaining visibility into the user actions by analyzing partial input can help detect usability issues, and profile users to predict their likelihood of completing the task successfully.

If their partial input has been captured and stored persistently, users are able to restore the state of their tasks and continue from where they left off. Users may also get auto-completion suggestions, which are retrieved as their partial input is combined with the input of countless previous users attempting a similar task. Returning users may also get incentives to finally complete their task, or – on the contrary – as they demonstrate a renewed willingness to convert, get penalized with slightly higher prices.

"Companies are nowadays using customers to market, sell, and create products for them" [55]. The distinction between end users and beta testers is blurred whenever people struggle through a poorly designed UI, report bugs and discuss among themselves how to work around them or even risk their lives without any compensation[2] for their pain/effort as they are implicitly providing invaluable feedback to the software provider [48].

Ethical Violation:

Users often assume that their input is not processed (nor stored) before they explicitly issue a command to do so (e.g., by clicking a

[2]Users will definitely benefit from the opportunity of buying the next generation of products, much improved thanks to their unacknowledged contribution. See `Training Data Harvester`.

save button, submitting the form), thus they may not expect that their partial, private, and possibly incorrect input is already visible to the service provider, and might be used against them.

Drawing the line:

What are the assumptions of users regarding the privacy of their interactions with the system? Are the smart autocompletion suggestions generated locally on the same device, or is the user partial input sent unencrypted, keystroke by keystroke, across the Internet [39] and stored forever in the Cloud?

If the user input is collected to make a decision affecting the users themselves, will the decision be based exclusively on the submitted information, or also considering on how many typos, corrections, hesitations, false starts, and breaks the user took during the preparation of the submission package?

A stealthy input logger crosses the line because it metaphorically places every user behind a one-way mirrored wall, without users being aware that the whole software development, operations and analytics team watching them is making bets on which button they will press next.

Known Uses:

Major travel booking sites send an email roughly 24h after a search for an accommodation has been left unfinished. The email includes a few recommendations nudging the recipient towards restarting the booking process.

A/B testing is a common practice for performing comparative empirical evaluations of the performance of competing designs as they are applied to unsuspecting users [11, 48].

3 Monetization at all costs

Creating digital goods — arranging bits in a particular way — is an invaluable skill. “Who can afford to do professional work for nothing?” [29] There is no question that creators should be compensated for their time and the resources they invest into making their creations [43].

Over the years, many business models have been introduced to market and sell digital goods, often resulting in the intermediary platforms capturing most of the revenue for themselves [49] as the terms of ownership change from buying to renting (on the consumer side) and employees are reclassified as contractors (on the creator side).

Monetization at all costs prioritizes short-term financial profits over user satisfaction, developer reputation and long-term sustainability. “Although monetization is very business centric, there is also some technical knowledge required to make sure that monetization work well” [85]. Indeed it would not be possible without applying software engineering skills to build instances of the Aggressive advertiser, AD-blocker detector, Pay to win, Artificial scarcity hoarder, DRM rug puller, and Obsolescence planner dark patterns.

Dark Pattern 3 - Aggressive advertiser

Context:

While some applications offer ad-free paid subscriptions, most providers of free content rely on revenue from ads [77].

Problem: *How to maximize ad revenue?*

Solution: Increase the quantity of ads displayed as much as possible:

- Allocate most of the user interface real estate to display ads.
- As a response to user interactions, open pop up windows entirely filled with ads, so that the user must close them to be able to access the screen they have been interacting with.
- Interleave ads within user provided content. For textual content, insert an ad between every paragraph. For image slideshows, insert an ad between every slide. For video, do not allow users to start watching their video before being exposed to at least one video ad, then randomly interrupt the video with additional ads, or place banner ads over the underlying video.
- From time to time, play full screen advertisement videos and leave no option to the user for skipping them.
- On full-screen video or popup ads, place fake close buttons which trick users into clicking on the ad.

Consequences:

The number of ad impressions and possibly the click-through-rate will grow.

The reputation of the content provider will suffer. Ads are annoying distractions [8]: not only they waste time and resources, but also raise serious privacy concerns [106]. Users are kept under surveillance [108] so they can be targeted with the same ads over and over across different applications. Ads have been also used a vector for malicious attacks [18, 91].

There is a limit beyond which users will no longer tolerate the invasive presence of ads [31]. Beyond the limit users will either stop using the application or install an ad-blocker[3].

To avoid this reaction, the amount and proportion of ads should be gradually increased over time [6]. New users get an almost ad-free experience, while returning users are presented with more and more invasive ads.

Ethical Violation:

Attention is a scarce and limited resource. In the age of information overload, diverting attention towards unwanted, distracting and sometimes annoying ads is even more wasteful.

Drawing the line:

While providing free content is not economically sustainable, especially when facing growth in traffic, advertising is a common way to generate revenue, especially with popular content providers [24]. Once the recurring infrastructure costs and salaries have been paid thanks to ad clicks, is there a point where the obscene profits obtained by an aggressive advertiser no longer justify annoying the user population?

Known Uses:

While many applications initially succeed on their value proposition and high content quality, there comes a time when their ownership is transferred from the original developers to new management interested in recouping the cost of the acquisition in the shortest possible time. At this stage, the user experience tends to degrade as the focus of the application shifts from providing value to users to extracting ad clicks from them.

[3]See AD-blocker detector.

Dark Pattern 4 - AD-blocker detector

Context:

While some applications offer ad-free paid subscriptions, most providers of free content rely on revenue from ads [77].

Users tend to avoid clicking on ads as they are mostly interested about the content (or prefer to engage with the application features) [13].

With more and more invasive ads being aggressively displayed, users resort to ad blocking tools [94] to protect their privacy, ensure the content remains accessible to them and ad-filled application screens maintain a minimum level of usability.

Problem: *How to force blocked ads to be displayed?*

Solution:

Test whether the ad gets displayed in front of the user. If the ad has been blocked, 1) prevent the user from accessing the actual content; 2) tell the user to disable the ad blocker so that the content will be shown together with the ads.

Consequences:

Blocking users that employ ad blockers may backfire as users are driven away towards other content sources. Some users who can afford it may also be enticed to become paid subscribers with the promise[4] to keep their experience free from ads.

Native ads, a form of product placement where ads are deeply embedded within the content, may be more challenging to block as ads are served from the same source as the content provider and therefore also assuage privacy concerns.

Ethical Violation:

While blocking deceptive, manipulative or malicious ads may be morally justified [103], some targeted ads may be simply informative, entertaining or even beneficial for users that discover exactly the right recommendation at the right time. Users should remain free to choose whether or not to be exposed to ads, unless they are contractually obligated to view ads in exchange for receiving the corresponding content.

Drawing the line:

The ad-blocker detector crosses the line because it takes away the user's freedom to choose whether or not to be exposed to ads. Some ad blocking tools offer powerful filters that can be customized. Thanks to them, users can freely decide on which site to block or display ads. For a fee, the authors of some ad blocking tools will whitelist some ads and display them instead of blocking them by default. In some cases, this revenue stream is shared with the users to which the ads are shown.

If our collective attention is valuable, the ongoing struggle between advertisers and ad-blockers may result in recognizing the right for everyone to freely 1) decide whether to be subjected to ads or not and 2) determine what is the price for selling a span of their precious attention.

Known Uses:

Most online news sources detect and also block attempts to separate the news content from the ads. They may employ soft ad blocker detectors, which partially hide the content from view or prevent users from scrolling to read it completely. Hard ad blocker detectors will simply stop delivering the article content.

Ad-blocking has been called one of the greatest boycotts in history, leading to the emergence of ad blocker detectors, which will lead to further developments in *stealth* ad blocking tools, or the careful engineering of tools to block both ads as well as ad blocker detectors themselves.

[4]See Sneaky terms degrader.

Dark Pattern 5 - Pay to Win

Context:

Gamers have limited time to play. Gamers enjoy winning a game [26] and like to compete to see themselves at the top of the player rankings.

Game development projects need to be economically sound. Piracy affects game developer revenues, as gamers obtain access to games without paying for them.

Problem: *How to make gamers pay to play games?*

Solution:

Give away games for free, but make it almost impossible to advance in the game by solely relying on patience, luck, and skill.

Design a game featuring an "in-game economy" with a virtual currency and a marketplace of power ups that can be obtained randomly (i.e., loot boxes), over time (e.g., gold farming) or by payments. Define an exchange rate between the virtual game currency and the player real-world currency and implement the means to transfer funds into the game (but not the other way). Provide volume discounts when exchanging larger sums.

By default, make gamers progress towards victory slowly. Introduce the opportunity for players to purchase the means to speed up the progress of the game for a fee (i.e., pay to accelerate [93]).

Similarly, the difficulty of the game should gradually increase until players are no longer able to succeed without purchasing the required power-ups that will tilt the playing field in their favor. For example, make some advanced levels almost impossible to win without purchasing the means to do so.

In case of multi-player games, sell power-ups which provide an advantage over other players.

For games that have a complex, multi-level structure, sell access to the final level(s).

For games where players can gather resources, sell more capacity to store and carry forward rewards, which would be lost otherwise due to artificially introduced storage size limits.

Consequences:

Gamers do not have to make an upfront investment before they can try to play a new game.

Game developers need to finely tune the speed or difficulty of the game so that players do not leave and give up because it takes too long to win it. Still, if the game runs too quickly or it is too easy, players have no incentive to purchase power ups from the game developers.

While some casual players may be content to follow the game at its default speed, and build up their skills over time to overcome the game challenges, the game developer can break even if a sufficient number of hardcore gamers keep paying to win [41].

Ethical Violations:

Players should not be nudged towards paying to win as they face the alternative of spending countless hours "grinding" at menial "gold farming" tasks to earn game credits.

Giving paid access to sources of randomly generated power ups has been deemed a hidden form of gambling [4].

Players often feel frustrated and cheated if during the game they discover that they can finish the game they started for free only by paying to enter the final level.

Some very young players [51] may not even realize that they are spending someone else's funds so that they can win the game.

Drawing the Line:

Is it still a fair game if players can pay to win? And what is the right price for victory? And where is the fun in pitting players who bought their way up against players who simply like to play?

Most in-game payments are intentionally kept small with micro-transactions, but there can be also some highly expensive and as a consequence very powerful speedup items for sale.

Games exploit gamification techniques [5] to engage players and turn their occasional game playing activities into a hard-to-resist habit [52]. This will lead to an irresistible urge to keep playing and the price of ending the game will seem like a small ransom to pay to finally quit their addiction.

Known Uses:

20 pay-to-win games have been listed here [80].

Dark Pattern 6 - Artificial scarcity hoarder

Context:

Digital goods (i.e., books, music, videos, software) can be replicated for negligible costs. Storage space for digital goods is virtually infinite. The time required to transfer copies of digital goods across different storage devices is becoming instantaneous. The vast abundance of digital goods overwhelms the very limited attention span which consumers can dedicate to them.

Problem: *How to intentionally restrict the availability of digital goods to manipulate their price and increase profits?*

Solution: Many attempts have been made to thwart the fundamental properties of digital bits so they behave like physical atoms [67]:

- Bind digital goods with physical ones (such as copy-protection devices).
- Require owners to register and identify themselves to obtain access to only the digital goods they are currently entitled to.
- Introduce digital rights management (DRM [30]) to create a secure path between the digital storage and the output devices used to render the digital goods in front of their "owner" sensory organs.
- Make digital goods expire and fade away after a certain date[5].
- Strictly limit the number of times digital goods can be accessed.
- Tie the digital goods to a specific geographic location from which they can be accessed.

[5]See Obsolescence planner.

Consequences:

When ownership of digital goods is tied to ownership of physical ones, both share the same limitations: they need to be manufactured, stored, packaged, shipped and take time to be delivered; also they can be damaged, lost or stolen. However, these limitations are intrinsic only to physical goods.

Although physical goods can be bought and enjoyed anonymously and privately, digital goods are typically bound to the identity of their owner that needs to be registered at the time of purchase and authenticated every time the goods are accessed.

While not all DRM schemes require user authentication, DRM will block some fair usage scenarios. DRM protections may also malfunction and lock out the rightful owner[6].

Even if physical objects wear out, digital goods can be preserved longer than their underlying physical substrate: the original bit arrangements could theoretically survive forever. However, the media can be decoded, the text can be rendered, the software can run only if the corresponding platform is kept functioning.

Some physical devices rely on consumable, limited resources that expire and need replacement (e.g., the fuel of an internal combustion engine, the ink of a printer). Digital goods do not have such limitations and can be enjoyed indefinitely.

In a similar way, physical objects exist in their geographic location, and can sometimes cross borders, along with their owners. While the traditional ownership relationship is not affected by the geographic location of the owner and their property, the same does not seem to hold for geo-fenced digital goods, which can only be enjoyed while the owner is located in certain places [22].

Ethical Violation:

Creating artificial scarcity in digital goods (e.g., by deceptive marketing practices which generate the illusion of exclusivity, with limited releases, editions, or timed drops) misleads and exploits consumers by inflating prices unjustifiably and exacerbating economic inequality. Only those who can afford it have access to resources which are becoming more and more essential. The perception of scarcity can create a sense of urgency (e.g., fear of missing out) leading consumers to impulsive purchases, which can in turn promote unhealthy behavior leading to financial stress.

Drawing the line:

If the entirety of the whole human knowledge will soon fit on a USB memory stick, why shouldn't everyone be able to carry one around in their pocket? What should be a fair market price for it? Who would deserve the privilege of setting such price? "Nobody bothers to create property for some resource that lies around in abundance" [81].

The artificial scarcity hoarder crosses the line as it attempts to make a "limited edition out of unlimited" [74] going against the very nature of digital goods: "even assuming that it were possible to totally prevent piracy, the infinite durability of digital goods would still undermine the profits of the digital goods industry" [79].

Known Uses:

Non fungible tokens (NFTs) are often released in exclusive digital art markets where delivering a veneer of digital scarcity induces a significant consumption of actually scarce natural resources [84].

[6]See DRM rug puller.

Dark Pattern 7 - DRM RUG PULLER

Context:

Digital Rights Management (DRM) have been introduced to ensure exclusively authorized access to digital goods [25]. DRM software relies on remote, Cloud-based services to perform authorization and authentication, whose availability is critical to ensure continued access to the protected digital goods.

Problem: *How to make users pay again for DRM-protected digital goods they already have bought?*

Solution:

DRM-protected players should disallow access when the Cloud backend they depend on is not available.

Shut down the DRM backend in the Cloud.

All players cannot grant access since they are no longer unable to perform the user authorization.

Consequences:

Owners have to buy "their" digital goods one more time under another, more up-to-date DRM scheme.

The concepts of ownership and property do not work as expected with DRM. Access to purchased digital goods is only possible as long as the DRM system works. When the DRM backend is no longer functional, "owners" of digital goods will no longer be able to access them (even if the bits they paid for are still stored on their devices). Circumventing DRM technology remains illegal even after it fails.

Ethical Violation:

From an economic perspective, it is impossible that the price paid for purchasing the digital good can subsidize the costs of running DRM systems forever, or at least during the whole lifetime of the digital good.

The DRM rug pull happens when the DRM system is intentionally shut down, even before it is no longer economically viable to keep operating it.

Drawing the line:

As technology rapidly evolves, DRM needs to keep up as well as to continue supporting legacy devices[7].

Honest people dislike being treated like potential thieves. DRM protected media imposes a tax both in terms of usability, interoperability and complexity, which may prevent casual piracy but also make the customer experience worse and discourage sales.

Neither legislation nor technological roadblocks will deter organized criminals who seek to bypass the DRM access control mechanisms [21]. After the DRM rug pull happens, circumvention tools may be the only viable means left to access the media.

DRM works exactly as intended by treating everyone as a potential attacker, including the rightful "owner" of the digital media, who remains so only until before the rug is pulled.

Known Uses:

Popular music players using proprietary DRM-locked formats eventually stopped working without offering any path for their users to migrate the digital music files they purchased so they could be played with alternative applications.

[7]See OBSOLESCENCE PLANNER

Dark Pattern 8 - OBSOLESCENCE PLANNER

Context:

There are unlimited resources available. It is possible to precisely control the durability of manufactured goods. While entropy is a physical phenomenon that leads to damage and decay, software entropy reflects the same destiny of digital goods [12].

Problem: *How to keep profiting from a saturated market?*

Solution:

Create a need for unnecessary upgrades by artificially setting a controlled bound on the operating lifetime during which digital goods can be used.

This bound can be set explicitly and directly, for example, by hard-coding an expiration date on a specific software release. Often it is set indirectly, for example, by raising the minimum hardware requirements of an operating system release so that it excludes older hardware configurations.

Consequences:

When it stops working, it needs to be replaced. Repair and recovery are not an option in a throwaway society [14].

More in detail, when the application is no longer compatible, the operating system needs to be updated. When the operating system update cannot be installed, the hardware needs to be upgraded. When the hardware needs to be upgraded, by design, it is not possible to perform specific component upgrades (e.g., adding more memory or storage space), therefore the entire device needs to be replaced with a new one.

Ethical Violation:

Entropy does not need our help. Resources are limited and eventually there will nothing left to mine but our own trash.

Drawing the line:

Any device with a non-replaceable battery will become a brick.

Any device with proprietary, no longer supported software or operating systems will become a brick.

Any device which communicates using non-standard and non-backwards-compatible protocols will become a brick.

Any device which depends on the Cloud (for authorization, for storage, for computation offloading, for boosting its intelligence, for delivering firmware updates) will become a brick.

While the context surrounding a system will inevitably change, such changes happen outside the control of the architect of the system. The obsolescence planner crosses the line by introducing specific obsolescence triggers and intentionally embedding them within the system so that its performance degrades over time and before its time.

In the long run, free and open technology outlasts proprietary software and data formats [35].

Known Uses:

The limiting factor can be placed in the hardware itself (e.g., crashed disks, batteries which no longer charge, failed ports, broken keys and switches, worn out connectors). If the hardware still works, it may become too "slow" to handle the bloat of more recent software stacked on it. Or simply its operating system may reach its end of life. Or the programming language compiler no longer targets an old operating system version. As a consequence, critical software applications are no longer receiving updates.

Noteworthy are Web browsers, which strive to keep backwards compatibility when calling ancient Web servers. However, even ever-green browsers tend to discontinue support for old operating system versions relatively quickly.

4 Digital Fraud

There is an inordinate amount of trust users put into their digital devices, making life easy for digital fraudsters to take unfair advantage of them. While there are many possible online scams, worthless cryptocurrencies, and security breaches performed by global ransomware gangs – to name a few – in this section we focus on two specific types of frauds: impersonation and the so-called "bait and switch".

Impersonation involves assuming someone else's identity. Fraudulent impersonation uses it to deceive others to gain access to restricted information, to inflict damage to someone's reputation, or simply for financial gain. A Cybersquatter is an impersonator found at the boundary between the digital and the real-world identity, sometimes one character or one pixel away from the true one.

Bait and Switch frauds in the digital world work exactly like in the real world: they lure a user with the promise of something attractive (too good to be true) and after the user takes the bait, they switch to something less desirable or even harmful. In this paper we present how bait and switch is applied at the business level – where the initially appealing terms of service (ToS) are gradually changed into a more expensive, more difficult to cancel, or simply disappointing deal (Sneaky terms degrader) — and at the technical level (interoperability breaker).

Dark Pattern 9 - Cybersquatter

Context:

Some users still type URLs into browser address bars. A few blindly trust the links they click on. Others prefer to scan QR codes – now ubiquitous – with their phone cameras.

Problem: *How to take advantage of users who trust the mistaken URLs they open?*

Solution:

Typo-squatting, or URL hijacking, targets people that accidentally mistype a website address directly into their web browser URL field [65]. It also targets developers who mis-type the name of the packages they install as dependencies in their projects [95]. More recently, a new variant of these attacks targets non-existing package names hallucinated by LLM code generators.

Phishing attacks rely on users who trust the links they receive in malicious email messages. Likewise, Qishing attacks work if users trust that any QR code they scan will bring them to a safe place. Since QR codes are not readable by humans, it is possible to replace them without human noticing the difference – unlike with typos introduced in Internet domain names.

Consequences:

Dealing with typosquatters increases the costs of domain name registration: domain names should not only be registered for all top-level domains but also for all possible common variations of the name when mistyping it. This way user who mis-type the domain name can be redirected back to the correct spelling. While registering a single domain may be relatively cheap, protecting its many variations can significantly impact recurring DNS-related costs.

Links embedded in email messages can be checked and filtered to ensure they are referring to a legitimate target. Note that while doing so, the original links in email messages are often replaced so that users clicking on them can be warned they may be opening a malicious site, also making it convenient to track who has a tendency to click on such links and when they last did so. In case of bad consequences, it's good to have someone to blame.

While QR codes printed on stickers that overlap with existing QR codes may indicate that the code has been upgraded or corrected since the original one underneath was printed, these should be treated with suspicion. If a QR code is printed next to a URL address, it is possible to decode the QR code and check whether it is consistent with the human readable URL before opening it. Still, such consistency does not imply safety.

Ethical Violation:

Typosquatting involves registering domain names that are typographical variations on names in which the malicious registrant lacks any legal right. Additionally, the intention is to mislead and confuse typo-prone users, to damage the reputation or simply extort money from the rightful trademark owner.

Cybersquatters exploit the tendency of users to blindly trust QR codes, especially when these are placed in a mostly harmless context.

Drawing the line:

What is the goal of the cybersquatter? Do they simply stand in the shadow of high traffic properties and benefit from the trickle of mistaken visitors? Do they attempt to trick mistaken users into revealing their credentials by impersonating a trusted storefront? Or do they benevolently help lost users find the right way?

Known Uses:

This study [65] crawled more than 285 000 typo-squatting domains targeting the top 3 264 .com sites.

Web browsers contain blacklists of URLs with typos which help to mitigate some of these attacks, assuming the editors of such blacklists are trustworthy and swift with their updates.

Dark Pattern 10 - Sneaky terms degrader

Also known as: **Darth Vader deal-maker**: "I am altering the deal. Pray I do not alter it any further" [47].

Context:

End users are interested to run "their" software, access "their" content. End users quickly scroll through lengthy licence agreements so they can click on the `Accept` button and start working. Very rarely they open and read privacy policy statements or complex terms of services [10].

Problem: *How to comply to the letter with the requirement to inform users about changes in the agreement while making it unlikely for them to withdraw if they no longer agree?*

Solution:

While users need to be notified that a change has occurred, they do not have to be explicitly informed about the specific details of the change. More in detail:

(1) Send a notification as late as it is legally possible to the users informing them that the terms have changed.
(2) Include a cursory summary of the changes (e.g., clarity of the wording has been improved, it's all in your benefit anyway).
(3) Share a link to the new agreement text. Do not highlight which clauses have been modified or added w.r.t. the previous version of the agreement.
(4) Inform users that no action is required on their part, if they keep using the software, they implicitly agree to the new terms.

Consequences:

End users may or may not notice the notification. Some of the end users that do notice may open the link. They will be able to read the new version of the terms extensively, but without any means to compare it with the previous one and check what actually changed. Users have invested their time in learning how to use the software, entered all their data into it and may not easily find a replacement within the short time frame mentioned in the notification. Also, users are given no option to keep being users of exactly the same software under the previous terms.

Ethical Violation:

While an entire ToS agreement may be too long to read carefully, changes are usually much shorter, so why not present at least the changes so that they are easy to notice and understand?

Drawing the line:

Introducing changes that negatively affect or harm users should not be done in a deceptive way, by attempting to hide the changes, by downplaying their impact, and without disclosing possible alternatives. While default acceptance (opt-out) is appropriate for a rapid transition towards improved terms, users should be given ample time to study the changes so that they become aware of their implications. Then, users should be required to explicitly give their consent (opt-in) if the new terms do not play in their favour.

The sneaky terms degrader crosses the line by downplaying the impact of changes and making it more difficult than it should be to understand what actually changed.

Known Uses:

"'Enshittification' is coming for absolutely everything" [20].

Dark Pattern 11 - INTEROPERABILITY BREAKER

Context:

External consumers built a thriving client application ecosystem, which brings more and more users and traffic through the API. Users can filter content and customize how it is displayed by picking the corresponding client application, leading to an inconsistent but flexible user experience. Users may also develop automated scripts that go beyond the API integration scenarios originally envisioned by the service provider.

Problem: *How to ensure full control over the client application user experience by the API service provider?*

Solution:

Service providers regain control over their client ecosystems by:

blocking – if API keys [107] are in place, simply revoke the ones of rogue clients which will stop functioning.

breaking – release a new API version which breaks most existing versions of client applications but release in lock-step patches to keep the officially sanctioned clients compatible.

***benching* (or ghosting)** – modify the API backend to stop sending data when responding to unwanted clients, without breaking them (i.e., no errors will be reported to users, which will be left to stare at empty screens).

Consequences:

After the intentionally backwards incompatible API is released, users can no longer work with their client applications which supported filtering unwanted content, nor they can modify how to render the content retrieved from the API to suit their needs by switching between alternative front-end applications.

Developers of client applications or scripts may always update them to recover compatibility with the latest API version. This has a non negligible cost, which forces some client developers out of the Sisyphean race to maintain the compatibility of their clients.

Users of client applications which consume feeds from multiple APIs may not realize they are no longer receiving content sourced from the API whose interoperability has been – silently – broken.

Ethical Violation:

The balance of power between API service provider and client developers shifts as a growing set of clients depends on the API, which becomes more and more popular and irreplaceable. Client application developers flock towards a given API under the assumption that it provides an attractive but stable and reliable platform. Their significant investment in building software on top of the API can be quickly nullified if the underlying API simply breaks compatibility or arbitrarily invalidates their API keys. This leads to spending more and more time and effort dealing with API changes as opposed to developing innovative application features.

Drawing the line:

The arc of software evolution naturally bends towards breaking interoperability. It requires extra effort to maintain backwards compatibility and design APIs with forwards compatibility in mind. Breaking interoperability may help to promote strict standard compliance or ensure that misbehaving clients are unable to connect.

Interoperability is what keeps software architectures open. It allows to compose APIs in unexpected and innovative ways. After taking advantage of interoperability to gain traction, the interoperability breaker crosses the line when it begins to build a walled garden to exert more and more control over who is allowed to access the API and what they can or cannot do with it.

Known Uses:

Major social media platforms, after an initial phase offering open APIs to attract interest, have started to exert a very tight control over their clients. For example, disallowing ad-blocking or temporal sorting of feeds.

Major operating system vendors, after an initial phase focused on developing a rock-solid API to attract application developers, have started to develop their own competing client applications.

5 Ranking Manipulation

Users have come to expect their software to be functionally sound. They trust that given their valid input, the correct output is returned. For example, if the software claims to perform a database lookup, users assume that the result is a pure reflection of the content of the database to which the given sorting criteria has been applied. If the software is meant to compute the nearest location of a given shop or restaurant, users expect that the result solely depends on their position, the chosen mean of transportation, and the presence of traffic delays. The payment of a fee to be prominently listed, or the presence of a suspiciously large number of very positive reviews, should not affect the objective criteria used by the software to make ranking decisions.

The ranking manipulation dark pattern challenge these – possibly naive, but still valid – assumptions on how search results are provided. When ranking decisions are crowd-sourced, the Fake review generator will automate the production of reviews to improve or degrade the placement of a given listing. Alternatively, a Search ranking kickbacker will simply sell access to highly ranked spots while pretending that the results are still relevant and related to the search query.

Dark Pattern 12 - Fake review generator

Context:

Potential new customers have not yet directly experienced the quality of a commercial offering. Existing customers publish reviews sharing their positive or negative experience with a given product or service provider.

Problem: *How to influence the aggregate review scores?*

Solution:

Perform a shilling attack [89] developing software which automates the following steps:

(1) Register a large number of user accounts in the review-based recommender system.
(2) Generate (e.g., using a large language model [2]) generic but credible, positive or negative review texts.
(3) Slowly and gradually post the reviews from different user accounts using many different devices.

Consequences:

The number of fake reviews should be sufficient to significantly move the aggregated review statistics in the desired direction ("push or nuke" [92]): positive for the products whose reputation should get a boost, negative for the ones of competitors.

Post reviews from different accounts to make it more difficult to filter fake reviews once their author has been identified as untrustworthy [105]. Do not post all fake reviews at the same time, again to make it more difficult to detect and filter them based on their timestamp. For the same reason, use different devices connected to different networks to make it more challenging to correlated the fake reviews with the IP address of their source.

When it is not possible to automatically post a sufficiently large number of fake reviews, bribe users into posting them and create a marketplace for recruiting users willing to post reviews on someone else's behalf [37].

Ethical Violation:

The wisdom of the crowds on which the recommendation system relies assumes that review authors are honest [44]. Posting fake reviews is a deceptive practice which may give an unfair competitive advantage, erode trust, and damage reputations.

Casual users may be swayed by the aggregated review scores and possibly make sub-optimal or harmful choices. Careful users who inspect individual reviews may become suspicious of their generated content and lose trust in the vandalized review-based recommendation system.

Drawing the line:

Asking some of your friends to give a boost to your latest release with their reviews is one thing. Bribing strangers to post reviews intended to bring down the ratings of your competitors is another.

The fake review generator crosses the line thanks to its automated software carefully engineered to flood the system with massive amounts of fake reviews, which can tilt the balance either way.

Known Uses:

New entrants often struggle due to a lack of reviews. They can apply this pattern to purchase fake positive reviews to break the ice and manufacture a good reputation for themselves [36].

Both major mobile app stores are affected by fake reviews [62].

The U.S. Federal Trade Commission announced a proposed rule banning fake reviews and testimonials on June 30, 2023 [28].

Dark Pattern 13 - Search ranking kickbacker

Context:

Search engine ranking algorithms are difficult for publishers to reverse engineer in order to predict how high a given content item (e.g., a product, store location, Web service API) will be placed in the ranking [54]. Also, it takes too much effort to keep up and react to frequent changes in how search engines rank results [16, 61].

Users trust the integrity of search engines: the most relevant and most timely results will be returned for a given query. Most users only follow the top recommendations of search engines, few scroll down looking for more results, and even fewer continue reading further results placed on the next pages [9].

Problem: *How to ensure arbitrary content is ranked in the top search engine results?*

Solution:

Users should still believe they see results organically ranked on relevance, while publishers get to control the actual ranking.

More in detail:

(1) Let publishers obtain a favorable ranking by paying a fee to the search engine.
(2) Display all search results uniformly regardless of whether their rank was organic or paid for.

Consequences:

Search engines get to levy a tax on publishers not only for promoting their content but for simply including it in the search results.

Giving priority to the payment of a kickback over the actual content relevance may boost irrelevant results to the top.

Search results cannot be anymore sorted correctly according to different criteria (e.g., relevance, freshness, price) since the same promoted results are stuck on top of the list.

Ethical Violation:

The integrity of the search engine is compromised. Users are deceived since they have no way to distinguish relevant results from the sponsored ones. Publishers of relevant content who cannot afford to pay to play become irrelevant.

Drawing the line:

Everything is fine as long as organic results can be somehow clearly distinguished from sponsored ones. There should be no doubt whatsoever for users about why a certain result has been placed on top. And if anyway users have a tendency to click on the first result, it's because they find it relevant for them, or maybe they are just feeling lucky.

Publisher named X – who already needs to worry about Cybersquatters taking ownership of similar names X' – should not have also to outbid competitors with a completely different brand name Y who often can afford to pay more to be listed as the top result for searches about X.

The search ranking kickbacker crosses the line because it betrays both its users by listing sponsored results disguised as organic ones and its publishers as it is willing to sell their correct, high-ranking placement to the highest bidder.

Known Uses:

Many freely accessible search engines prioritize revenue over relevance to rank their results [57].

Popular e-commerce sites let manufacturers enter bids for highly placed results as they run auctions for every user search [78].

6 Unethical Artificial Intelligence Practices

Artificial intelligence has recently made a big impact in software engineering [58, 73], raising questions about the continued need for programmers in the future [1, 98], the rapid propagation of technical debt through generated code [3], the viability of the past education methods on the next generation of software developers [76], as well as the usual lack of explainability, the presence of bias, and potential for misuse.

The dark patterns in this category cover three unethical AI practices: the unauthorized collection of vast amounts of training data (Training data harvester), the precarious attempts to replace humans (Bot Pretender), and the release into the wild of novel kinds of harmful software applications (Deceptive Deepfaker).

Dark Pattern 14 - Training data harvester

Context:

Statistical machine learning methods rely on large amounts of data for training purposes. Access to high quality curated data sources is expensive. The World Wide Web is a freely available source of massive amounts of textual and multimedia content in many different natural languages. While the Web is easy to crawl, in many cases, its content is not in the public domain. With the exception of fair use, copyright law [82] protects the rights to reproduce, prepare derivative works, distribute copies, digitally transmit, give a public performance or display of the copyrighted work.

Problem: *How to amass a sufficiently large quantity of training data suitable for a variety of applications?*

Solution:

Crawl the Web and feed its entire content – without checking the terms and conditions of each source, nor obtaining the proper authorization, and ignoring the `robots.txt` convention – to the machine learning algorithm.

Proceed to fine-tune the foundational (or pre-trained) model for specific tasks.

Before releasing the fine-tuned model perform extensive testing and add safeguards for quality control.

When the Web runs out of public pages to crawl, additional data can always be found in private repositories, internal databases or personal file systems [69].

Consequences:

The machine learning model can be trained with more and more datasets, without having to afford to pay the price for humankind's effort to create those datasets.

Copyright law is being tested on whether a model trained on vast amounts of copyrighted material can be considered as an infringing derivative work or a fair use exemption should be granted [104].

Some of the output generated by models is published back on the Web itself. The next generation of models will be trained based on data polluted by the output of previous model versions, eventually leading to model collapse [90].

Ethical Violation:

Given the complex and indirect relationship between the model, its output and the original content used for training, the authors of the training data are usually not compensated, nor cited, nor acknowleged [75, 101].

The cost of retroactively adding safeguards seems to be smaller than the cost of curating the vast amounts of training data upfront. The significant effort invested in authoring the original training material is disregarded.

Drawing the line:

The world wide web was invented as a common publishing medium for sharing humanity's knowledge. After a phase in which the web became programmable and provided the platform for building the universal client, the latest evolution of the web is now to act as the largest source of training data. While the HTTP status code `402 payment required` is still reserved but not yet defined, nor implemented by any Web browser, there are many valuable knowledge and news sources protected by paywalls.

The training data harverster crosses the line as it willfully ignores the original purpose for which the data was published and disregards existing conventions which limit automated crawlers.

Known Uses:

Most major large language models have been trained out of large amounts of copyrighted works, resulting in ongoing lawsuits.

As recently mentioned by a former CEO of a major AI technology player: “if nobody uses your product, it doesn't matter that you stole all the content” [38].

Dark Pattern 15 - Bot pretender

Context:

Artificially intelligent bots are being introduced in more and more challenging applications. Machine learning models can estimate the confidence of their inferences. There exist a crowd of reliable, and possibly remote human workers.

Problem: *How to transparently deal with limitations or failures of artificial intelligence bots and avoid accountability?*

Solution:

Send the user command prompt to the bot.

If the bot fails to parse it or generates a low confidence reply, send the user command prompt to the human worker [50].

Display the response to the user without mentioning that it was (or may have been) produced by a human worker.

Human workers can be dynamically hired from crowdsourcing platforms, further distancing them from the user they are interacting with, while pretending to be a bot.

Consequences:

Since the same communication channel is used when interacting with the bot and the human, users cannot tell them apart and will credit the bot for the intelligent responses originated from the human.

Still, users may perceive an unexpected delay or a variation in the usual response time in case the human takes over. This can be masked by randomly slowing down the bot or applying this pattern to non-interactive long-running tasks, which are performed after the chat session is completed.

Human crowd-workers receive only the difficult cases, the ones which could not be handled by bots.

Ethical Violation:

Bots lack agency and responsibility. Humans instead should be held accountable for their words or actions. A human masquerading as a bot may do so to attempt to avoid accountability.

From the users perspective, there is a fundamental lack of transparency, leading to deception. Also, user expectations when dealing with bots are set differently than when interacting with humans [40].

Drawing the line:

How often does the human need to take over? Will it work at all without a human hidden inside the bot? Will humans need to be kept in the loop, but only “just in case”?

Given the impedance mismatch between artificial intelligence and human intelligence, can humans actually act as effective and reliable supervisors of bots [32]? Will humans be always capable to accurately catch and properly override every mistake? Or by staying in the loop humans bring back the bias that was supposed to have been removed from algorithmic decision making?

Known Uses:

The original mechanical turk chess playing robot; the metaphor also fits well the digital version delivering access to humans-as-a-service [42].

A fleet of ”self-driving“ driverless robotaxis was remotely supported by 1.5 operators/vehicle [64].

Dark Pattern 16 - Deceptive Deepfaker

Context:

It has become possible to realistically synthesize or replace human traits (e.g., faces, voices) on digital media (pictures or videos).

Problem: *How to deceive viewers with misinformation which appears to be authentic?*

Solution:

Obtain a sufficiently long sound bite or a picture of the target to be impersonated without their permission.

Record a video portraying the action that should be performed, or write a text with the speech that should be said.

Use a deep learning tool to apply the human traits of the people to be impersonated to the video or perform a text to speech translation using a synthetic voice mimicking the one of the target.

Release the deepfake media without stating that the content has been artificially generated.

Consequences:

As deepfakes get more and more difficult to detect, digital media can no longer be fully trusted to offer a faithful depiction of factual reality [33].

Complex watermarking and provenance mechanisms need to be developed to ensure the authenticity of digital media [99]. Building more technology to address technology’s shortcomings starts an arms race with no winners.

Ethical Violation:

“Protection against the manipulation of hyper-realistic digital representations of our image and voice should be considered a fundamental moral right in the age of deepfakes.” [17].

Drawing the line:

While deepfake video production is not a software engineering practice in itself, the software applications to produce deepfake videos have been designed, developed, tested, and released to production following common software engineering practices, which in this case have been directed towards the production of deeply unethical software.

Known Uses:

Blackmail, intimidation, revenge porn, political messaging [7].

7 Conclusion

The dark patterns collected in this paper are used by unethical software engineers to develop specific software applications which intentionally satisfy harmful requirements, such as privacy violation, financial loss, depletion of limited resources (including waste of time), impersonation, fraud, erosion of trust, reputation damage, copyright infringement, worker exploitation and political manipulation.

Without software (and deep learning algorithms) there would be no deceptive deepfake videos. It would be impossible to harvest data at a scale sufficient to train foundational machine learning models with it. It would be too expensive to run millisecond-long auctions to gather bids for gaining higher search result ranks, or generate the amounts of fake reviews necessary for successful shilling attacks. Software is meant to be so flexible and easy to change, inducing into the temptation to gradually alter it further and further away from protecting the interests of its users: Software written to intentionally degrade the performance of an old but still functioning device as soon as a new version of a device is released for sale; Software which stops playing digital media "bought" before a certain date; Software which intentionally frustrates gamers into becoming gambling addicts; Software which detects the presence of software which detects the presence of software which detects the presence of software to block malicious ads; Software which spies on its users on behalf of their employer, of some government or major multi-national corporation.

While this paper focuses on the ethical implications of the patterns, it would also be possible to discuss their consequences from a legal perspective. Given that privacy regulations and AI legislation are a fast moving field, it could very well be that at some point, the dark patterns we identified will be deemed illegal.

Acknowledgement

The author would like to acknowledge the generous feedback shared by Olaf Zimmermann and the very helpful and patient shepherding of Eduardo Guerra. A big thank you goes also to the PLoP2024 workshop moderator (Joseph W. Yoder) and participants (Jim Episale, Zishan Rahman, Daniel Pinho, Hugo Sereno Ferreira, Rebecca Wirfs-Brock, Andrew P. Black) for sharing their invaluable feedback.

References

[1] Pekka Abrahamsson, Tatu Anttila, Jyri Hakala, Juulia Ketola, Anna Knappe, Daniel Lahtinen, Väinö Liukko, Timo Poranen, Topi-Matti Ritala, and Manu Setälä. Chatgpt as a fullstack web developer-early results. In *International Conference on Agile Software Development*, pages 201–209. Springer, 2024.

[2] David Ifeoluwa Adelani, Haotian Mai, Fuming Fang, Huy H. Nguyen, Junichi Yamagishi, and Isao Echizen. Generating Sentiment-Preserving Fake Online Reviews Using Neural Language Models and Their Human- and Machine-Based Detection. In Leonard Barolli, Flora Amato, Francesco Moscato, Tomoya Enokido, and Makoto Takizawa, editors, *Advanced Information Networking and Applications*, pages 1341–1354. Springer, 2020.

[3] Ali Al-Kaswan and Maliheh Izadi. The (ab)use of open source code to train large language models. In *Proc. 2nd International Workshop on Natural Language-Based Software Engineering (NLBSE)*, pages 9–10. IEEE/ACM, 2023.

[4] Tyler Amano-Smerling. Pay to play: The ethics of video game economics. *Viterbi Conversations in Ethics*, 5(1), 2021.

[5] Fernando R. H. Andrade, Riichiro Mizoguchi, and Seiji Isotani. The bright and dark sides of gamification. In Alessandro Micarelli, John Stamper, and Kitty Panourgia, editors, *Intelligent Tutoring Systems*, pages 176–186. Springer, 2016.

[6] Christina Aperjis, Ciril Bosch-Rosa, Daniel Friedman, and Bernardo A Huberman. Boiling the frog optimally: an experiment on survivor curve shapes and internet revenue. Technical report, SFB 649 Discussion Paper, 2014.

[7] Markus Appel and Fabian Prietzel. The detection of political deepfakes. *Journal of Computer-Mediated Communication*, 27(4):zmac008, 2022.

[8] Leif Azzopardi, David Maxwell, Martin Halvey, and Claudia Hauff. Driven to distraction: Examining the influence of distractors on search behaviours, performance and experience. In *Proc. 2023 Conference on Human Information Interaction and Retrieval*, CHIIR '23, page 83–94. ACM, 2023.

[9] Leif Azzopardi and Guido Zuccon. Two scrolls or one click: A cost model for browsing search results. In *Advances in Information Retrieval*, pages 696–702. Springer, 2016.

[10] Shmuel I Becher and Uri Benoliel. Sneak in contracts. *Ga. L. Rev.*, 55:657, 2020.

[11] Raquel Benbunan-Fich. The ethics of online research with unsuspecting users: From a/b testing to c/d experimentation. *Research Ethics*, 13(3-4):200–218, 2017.

[12] Gerardo Canfora, Luigi Cerulo, Marta Cimitile, and Massimiliano Di Penta. How changes affect software entropy: an empirical study. *Empirical Software Engineering*, 19:1–38, 2014.

[13] Chang-Hoan Cho and Hongsik John Cheon. Why do people avoid advertising on the internet? *Journal of Advertising*, 33(4):89–97, 2004.

[14] Tim Cooper, editor. *Longer Lasting Products: Alternatives to the Throwaway Society.* Gower, 2010.

[15] Reiner Creutzburg. The strange world of keyloggers-an overview, part i. *Electronic Imaging*, 29:139–148, 2017.

[16] Harold Davis. *Search engine optimization.* O'Reilly, 2006.

[17] Adrienne De Ruiter. The distinct wrong of deepfakes. *Philosophy & Technology*, 34(4):1311–1332, 2021.

[18] Michalis Diamantaris, Serafeim Moustakas, Lichao Sun, Sotiris Ioannidis, and Jason Polakis. This sneaky piggy went to the android ad market: Misusing mobile sensors for stealthy data exfiltration. In *Proc. SIGSAC Conference on Computer and Communications Security*, CCS '21, page 1065–1081. ACM, 2021.

[19] Virginia Dignum. *Responsible artificial intelligence: how to develop and use AI in a responsible way*, volume 2156. Springer, 2019.

[20] Cory Doctorow. 'enshittification' is coming for absolutely everything. *Financial Times*, February 2024. https://www.ft.com/content/6fb1602d-a08b-4a8c-bac0-047b7d64aba5.

[21] Christian D'Orazio and Kim-Kwang Raymond Choo. An adversary model to evaluate DRM protection of video contents on iOS devices. *Computers & Security*, 56:94–110, 2016.

[22] Nick Doty and Erik Wilde. Geolocation privacy and application platforms. In *Proc. 3rd SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS*, SPRINGL '10, page 65–69. ACM, 2010.

[23] Georgios Doukidis, Diomidis Spinellis, and Christof Ebert. Digital transformation - a primer for practitioners. *IEEE Software*, 37(5):13–21, 2020.

[24] Bruno Dyck and Rajesh V Manchanda. Sustainable marketing based on virtue ethics: Addressing socio-ecological challenges facing humankind. *AMS review*, 11:115–132, 2021.

[25] Edward W Felten. A skeptical view of DRM and fair use. *C. ACM*, 46(4):56–59, 2003.

[26] Guo Freeman, Karen Wu, Nicholas Nower, and Donghee Yvette Wohn. Pay to win or pay to cheat: How players of competitive online games perceive fairness of in-game purchases. In *Proc. Hum.-Comput. Interact. (CHI PLAY)*. ACM, 2022.

[27] Batya Friedman and David G Hendry. *Value sensitive design: Shaping technology with moral imagination.* Mit Press, 2019.

[28] FTC. Trade regulation rule on the use of consumer reviews and testimonials, 2023. 88 FR 49364 https://www.federalregister.gov/documents/2023/07/31/2023-15581/trade-regulation-rule-on-the-use-of-consumer-reviews-and-testimonials.

[29] Bill Gates. An open letter to hobbyists. *Homebrew Computer Club Newsletter*, 2(1), February 1976.

[30] Tarleton Gillespie. *Wired Shut: Copyright and the Shape of Digital Culture.* MIT Press, 2009.

[31] Daniel G Goldstein, Siddharth Suri, R Preston McAfee, Matthew Ekstrand-Abueg, and Fernando Diaz. The economic and cognitive costs of annoying display advertisements. *Journal of Marketing Research*, 51(6):742–752, 2014.

[32] Ben Green. The flaws of policies requiring human oversight of government algorithms. *Computer Law & Security Review*, 45, 2022.

[33] Joshua Habgood-Coote. Deepfakes and the epistemic apocalypse. *Synthese*, 201(3):103, 2023.

[34] T. Hansen and A. Melnikov. Rfc 8098: Message disposition notification, 2017.

[35] Masayuki Hatta. The right to repair, the right to tinker, and the right to innovate. *Annals of Business Administrative Science*, 19(4):143–157, 2020.

[36] Sherry He, Brett Hollenbeck, and Davide Proserpio. Exploiting social media for fake reviews: evidence from Amazon and Facebook. *SIGecom Exch.*, 19(2):68–74, November 2021.

[37] Sherry He, Brett Hollenbeck, and Davide Proserpio. The market for fake reviews. *Marketing Science*, 41(5):896–921, 2022.

[38] Alex Heath. Ex-Google CEO says successful AI startups can steal IP and hire lawyers to 'clean up the mess'. *The Verge*, 2024. https://www.theverge.com/2024/8/14/24220658/google-eric-schmidt-stanford-talk-ai-startups-openai.

[39] D.M. Hilbert and D.F. Redmiles. An approach to large-scale collection of application usage data over the internet. In *Proc. 20th International Conference on Software Engineering (ICSE)*, pages 136–145, April 1998.

[40] Pamela J Hinds, Teresa L Roberts, and Hank Jones. Whose job is it anyway? a study of human-robot interaction in a collaborative task. *Human–Computer Interaction*, 19(1-2):151–181, 2004.

[41] Kenton Howard. Free-to-play or pay-to-win? casual, hardcore, and hearthstone. *Transactions of the Digital Games Research Association*, 4(3), Oct. 2019.

[42] Lilly C Irani and M Six Silberman. Turkopticon: Interrupting worker invisibility in amazon mechanical turk. In *Proc. SIGCHI conference on human factors in computing systems*, pages 611–620, 2013.

[43] Adrian Johns. *Piracy: The intellectual property wars from Gutenberg to Gates.* University of Chicago Press, 2019.

[44] Radu Jurca and Boi Faltings. Truthful opinions from the crowds. *ACM SIGecom Exchanges*, 7(2):1–4, 2008.

[45] Stefan Kapferer, Olaf Zimmermann, and Mirko Stocker. Value-driven analysis and design. In *Proc. 29th European Conference on Pattern Languages of Programs, People, and Practices (EuroPLoP)*, 2024.

[46] Leo Kelion. Spy pixels in emails have become endemic. *BBC*, February 2021. https://www.bbc.com/news/technology-56071437.

[47] Hila Keren. I am altering the deal. pray i don't alter it any further. *Jotwell: J. Things We Like*, page 1, 2020.

[48] Ron Kohavi, Diane Tang, and Ya Xu. *Trustworthy online controlled experiments: A practical guide to a/b testing.* Cambridge University Press, 2020.

[49] Susanne Kopf. "rewarding good creators": Corporate social media discourse on monetization schemes for content creators. *Social Media + Society*, 6(4):2056305120969877, 2020.

[50] Pavel Kucherbaev, Alessandro Bozzon, and Geert-Jan Houben. Human-aided bots. *IEEE Internet Computing*, 22(6):36–43, 2018.

[51] Daria J Kuss and Mark D Griffiths. Online gaming addiction in children and adolescents: A review of empirical research. *Journal of behavioral addictions*, 1(1):3–22, 2012.

[52] Daria Joanna Kuss and Mark D Griffiths. Internet gaming addiction: A systematic review of empirical research. *International journal of mental health and addiction*, 10:278–296, 2012.

[53] Susan Landau. Making sense from Snowden: What's significant in the NSA surveillance revelations. *IEEE Security & Privacy*, 11(4):54–63, 2013.

[54] Jerri L Ledford. *Search engine optimization bible.* Wiley, 2nd edition, 2015.

[55] Bill Lee. *The hidden wealth of customers: Realizing the untapped value of your most important asset.* Harvard Business Press, 2012.

[56] Lawrence Lessig. *Code: And other laws of cyberspace.* Basic Books, 1999.

[57] Dirk Lewandowski, Friederike Kerkmann, Sandra Rümmele, and Sebastian Sünkler. An empirical investigation on search engine ad disclosure. *Journal of the Association for Information Science and Technology*, 69(3):420–437, 2018.

[58] David Lo. Trustworthy and synergistic artificial intelligence for software engineering: Vision and roadmaps. In *Proc. International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE)*, pages 69–85, 2023.

[59] Qinghua Lu, Liming Zhu, Jon Whittle, and Xiwei Xu. *Responsible AI: Best Practices for Creating Trustworthy AI Systems.* Addison-Wesley, 2023.

[60] Kevin Macnish. *The ethics of surveillance: An introduction.* Routledge, 2017.

[61] Ross A. Malaga. Worst practices in search engine optimization. *Commun. ACM*, 51(12):147–150, dec 2008.

[62] Daniel Martens and Walid Maalej. Towards understanding and detecting fake reviews in app stores. *Empirical Software Engineering*, 24(6):3316–3355, 2019.

[63] Arunesh Mathur, Mihir Kshirsagar, and Jonathan Mayer. What makes a dark pattern... dark? design attributes, normative considerations, and measurement methods. In *Proc. Conf. on Human Factors in Computing Systems (CHI)*, 2021.

[64] Tripp Mickle, Cade Metz, and Yiwen Lu. G.M.'s Cruise moved fast in the driverless race. it got ugly. *New York Times*, Nov 3, 2023.

[65] Tyler Moore and Benjamin Edelman. Measuring the perpetrators and funders of typosquatting. In Radu Sion, editor, *Financial Cryptography and Data Security*, pages 175–191. Springer, 2010.

[66] Vidhya Navalpakkam and Elizabeth Churchill. Mouse tracking: measuring and predicting users' experience of web-based content. In *Proc. Conf. on Human Factors in Computing Systems (CHI)*, page 2963–2972. ACM, 2012.

[67] Nicholas Negroponte. *Being digital.* Vintage, 2015.

[68] C Thi Nguyen. Echo chambers and epistemic bubbles. *Episteme*, 17(2):141–161, 2020.

[69] Liang Niu, Shujaat Mirza, Zayd Maradni, and Christina Pöpper. CodexLeaks: Privacy leaks from code generation language models in GitHub copilot. In *Proc. 32nd USENIX Security Symposium*, pages 2133–2150, 2023.

[70] Helena Holmström Olsson and Jan Bosch. Going digital: Disruption and transformation in software-intensive embedded systems ecosystems. *Journal of Software: Evolution and Process*, 32(6):e2249, 2020. e2249 JSME-19-0109.R2.

[71] Cathy O'neil. *Weapons of math destruction: How big data increases inequality and threatens democracy.* Crown, 2017.

[72] Rolf Oppliger. Certified mail: the next challenge for secure messaging. *Commun. ACM*, 47(8):75–79, August 2004.

[73] Ipek Ozkaya. Application of large language models to software engineering tasks: Opportunities, risks, and implications. *IEEE Software*, 40(3):4–8, 2023.

[74] Rachel O'Dwyer. Limited edition: Producing artificial scarcity for digital art on the blockchain and its implications for the cultural industries. *Convergence*, 26(4):874–894, 2020.

[75] Frank Pasquale and Haochen Sun. Consent and compensation: Resolving generative ai's copyright crisis. *Virginia Law Review*, 110:207–247, August 2024.

[76] James Prather, Brent N Reeves, Juho Leinonen, Stephen MacNeil, Arisoa S Randrianasolo, Brett A. Becker, Bailey Kimmel, Jared Wright, and Ben Briggs. The widening gap: The benefits and harms of generative ai for novice programmers. In *Proc. 2024 ACM Conference on International Computing Education Research - Volume 1*, ICER '24, page 469–486. ACM, 2024.

[77] Enric Pujol, Oliver Hohlfeld, and Anja Feldmann. Annoyed users: Ads and ad-block usage in the wild. In *Proc. Internet Measurement Conference (IMC)*, page 93–106. ACM, 2015.

[78] Tao Qin, Wei Chen, and Tie-Yan Liu. Sponsored search auctions: Recent advances and future directions. *ACM Trans. Intell. Syst. Technol.*, 5(4), January 2015.

[79] Thierry Rayna. Understanding the challenges of the digital economy: The nature of digital goods. *Communications & Strategies*, (71):13–16, 2008.

[80] Ty Richardson. Top 20 worst pay-to-win video games. *WatchMOJO*, 2023. https://www.watchmojo.com/articles/top-20-worst-pay-to-win-games.

[81] Carol M Rose. The several futures of property: Of cyberspace and folk tales, emission trades and ecosystems. *Minn. L. Rev.*, 83:129, 1998.

[82] Mark Rose. *Authors and owners: The invention of copyright.* Harvard University Press, 1993.

[83] Johann Rost and Robert L Glass. *The dark side of software engineering: evil on computing projects.* Wiley, 2011.

[84] Zahr K Said. Fables of scarcity in IP. *Frontiers in Research Metrics and Analytics*, 7:974154, 2023.

[85] Tamie Salter. Monetization. In *Technological and Business Fundamentals for Mobile App Development*, pages 133–157. Springer, 2022.

[86] Eugene Schlossberger. Engineering codes of ethics and the duty to set a moral precedent. *Science and Engineering Ethics*, 22(5):1333–1344, 2016.

[87] Eugene Schlossberger. *Ethical engineering: a practical guide with case studies.* CRC Press, 2023.

[88] Markus Schumacher, Eduardo Fernandez-Buglioni, Duane Hybertson, Frank Buschmann, and Peter Sommerlad. *Security Patterns: Integrating security and systems engineering.* Wiley, 2013.

[89] Anu Shrestha, Francesca Spezzano, and Maria Soledad Pera. Who is really affected by fraudulent reviews? an analysis of shilling attacks on recommender systems in real-world scenarios. *arXiv preprint arXiv:1808.07025*, 2018.

[90] Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson. The curse of recursion: Training on generated data makes models forget. *arXiv preprint arXiv:2305.17493*, 2023.

[91] Aditya K Sood and Richard J Enbody. Malvertising–exploiting web advertising. *Computer Fraud & Security*, 2011(4):11–16, 2011.

[92] Agnideven Palanisamy Sundar, Feng Li, Xukai Zou, Tianchong Gao, and Evan D. Russomanno. Understanding shilling attacks and their detection traits: A comprehensive survey. *IEEE Access*, 8:171703–171715, 2020.

[93] Thomas Tregel, Miriam Claudia Schwab, Thanh Tung Linh Nguyen, Philipp Niklas Müller, and Stefan Göbel. Costs to compete - analyzing pay to win aspects in current games. In Minhua Ma, Bobbie Fletcher, Stefan Göbel, Jannicke Baalsrud Hauge, and Tim Marsh, editors, *Serious Games*, pages 177–192. Springer, 2020.

[94] Nicholas Vincent, Brent Hecht, and Shilad Sen. "data strikes": Evaluating the effectiveness of a new form of collective action against technology companies. In *The World Wide Web Conference*, WWW '19, page 1931–1943. ACM, 2019.

[95] Duc-Ly Vu, Ivan Pashchenko, Fabio Massacci, Henrik Plate, and Antonino Sabetta. Typosquatting and combosquatting attacks on the python ecosystem. In *Proc. IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 509–514, 2020.

[96] Siqi Wang, Jun-Hwa Cheah, and Xin-Jean Lim. Online shopping cart abandonment: A review and research agenda. *International Journal of Consumer Studies*, 47(2):453–473, 2023.

[97] Olle Wästberg. Digitalization has changed the foundation of the democracy. In *Digital Transformation and Public Services*, pages 318–333. Routledge, 2019.

[98] Matt Welsh. The end of programming. *C. ACM*, 66(1):34–35, 2022.

[99] David Gray Widder, Dawn Nafus, Laura Dabbish, and James Herbsleb. Limits and possibilities for "ethical ai" in open source: A study of deepfakes. In *Proc. Conference on Fairness, Accountability, and Transparency*, FAccT '22, page 2035–2046. ACM, 2022.

[100] Norbert Wiener. Some moral and technical consequences of automation. *Science*, 131:1355–1358, 1960.

[101] Rui-Jie Yew. Break it 'til you make it: An exploration of the ramifications of copyright liability under a pre-training paradigm of ai development. In *Proc. Symposium on Computer Science and Law*, CSLAW '24, page 64–72. ACM, 2024.

[102] José P Zagal, Staffan Björk, and Chris Lewis. Dark patterns in the design of games. In *Foundations of Digital Games 2013*, 2013.

[103] Alexander Zambrano and Caleb Pickard. A defense of ad blocking and consumer inattention. *Ethics and Information Technology*, 20:143–155, 2018.

[104] Dawen Zhang, Boming Xia, Yue Liu, Xiwei Xu, Thong Hoang, Zhenchang Xing, Mark Staples, Qinghua Lu, and Liming Zhu. Navigating privacy and copyright challenges across the data lifecycle of generative ai. *arXiv preprint arXiv:2311.18252*, 2023.

[105] Haizhong Zheng, Neng Li, Minhui Xue, Suguo Du, and Haojin Zhu. Fake reviews tell no tales? dissecting click farming in content-generated social networks. In *Proc. International Conference on Communications in China (ICCC)*, pages 1–6, 2017.

[106] Yu-Qian Zhu and Jung-Hua Chang. The key role of relevance in personalized advertisement: Examining its impact on perceptions of privacy invasion, self-awareness, and continuous use intentions. *Computers in Human Behavior*, 65:442–447, 2016.

[107] Olaf Zimmermann, Mirko Stocker, Daniel Lübke, Uwe Zdun, and Cesare Pautasso. *Patterns for API Design: Simplifying Integration with Loosely Coupled Message Exchanges.* Addison-Wesley Signature Series (Vernon). Pearson, 2023.

[108] Shoshana Zuboff. The age of surveillance capitalism. In *Social theory re-wired*, pages 203–213. Routledge, 2023.