Università degli Studi di Bari Aldo Moro

Computer Science Department

PhD Programme in Computer Science and Mathematics

XXXVII Cycle

Scientific Disciplinary Area IINF-05/A

A framework for mitigating Trustworthy AI issues in practice

Coordinator:
Prof. Francesca MAZZIA

PhD Candidate:
Domenico GIGANTE

Tutor:
Prof. Maria Teresa BALDASSARRE

Co-Tutor:
Dr. Giovanni BRUNO

FINAL EXAM 2025

Abstract

The speed of the digitalization process, also due to the ongoing pandemic crisis, is leading to an exponential growth of threats to the privacy and security of software systems and a significant widening of the attack boundaries.

An additional significant trend is the rise of "AI-enabled systems," referring to software systems heavily reliant on Artificial Intelligence. This encompasses, for instance, autonomous and semi-autonomous vehicles, where vulnerabilities extend beyond application issues to include potential inadequacies of algorithms in accurately handling unusual and unforeseen behaviors. Moreover, vulnerabilities may stem from the integration of software systems with physical components, such as sensors.

The objective of this thesis is to propose processes and tools capable of operating in conventional and unconventional scenarios, supporting the software engineer in the development of secure systems.

Specifically, the objectives pursued are:

- Goal 1: Proposal of guidelines and tools to support the development process of conventional software systems.
- Goal 2: Proposal of guidelines and tools to support the development process of AI-enabled systems in a company;
- Goal 3: Formularization of a theoretical framework which encopasses all above guidelines and tools and provides support for both technical and non-technical stakeholders who participate in a corporate project.

Declaration by the author

This thesis is composed of my original work, and contains no material previously published or written by another person except where due reference has been made in the text. I have clearly stated the contribution by others to jointly-authored works that I have included in my thesis.

I have clearly stated the contribution of others to my thesis as a whole, including statistical assistance, survey design, data analysis, significant technical procedures, professional editorial advice, financial support and any other original research work used or reported in my thesis. The content of my thesis is the result of work I have carried out since the commencement of my higher degree by research candidature.

Publications included in this thesis

- [1] V. Barletta, G. Desolda, **D. Gigante**, R. Lanzilotti, M. Saltarella, From gdpr to privacy design patterns: The materialist framework, *Proceedings of the 19th International Conference on Security and Cryptography-SECRYPT*, 642-648, 2022

- [2] M. T. Baldassarre, V. Barletta, G. Dimauro, **D. Gigante**, A. Pagano, A. Piccinno, Supporting secure agile development: The vis-prise tool, *Proceedings of the 2022 International Conference on Advanced Visual Interfaces*, 1-3, 2022

- [3] **D. Gigante**, F. Pecorelli, V. Barletta, A. Janes, V. Lenarduzzi, D. Taibi, M. T. Baldassarre, Resolving Security Issues via Quality-Oriented Refactoring: A User Study, 2023 *ACM/IEEE International Conference on Technical Debt (TechDebt), co-located with the 46th International Conference on Software Engineering, ICSE 2023*, 82-91, 2023

- [4] V. Barletta, D. Caivano, **D. Gigante**, A. Ragone, A Rapid Review of Responsible AI frameworks: How to guide the development of ethical AI, *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering (EASE23)*, 358-367, 2023

- [5] M. T. Baldassarre, D. Caivano, B. F. Nieto, **D. Gigante**, A. Ragone, The Social Impact of Generative AI: An Analysis on ChatGPT, *Proceedings of the 2023 ACM Conference on Information Technology for Social Good (GoodIT23)*, 363-373, 2023

- [6] M. T. Baldassarre, **D. Gigante**, M. Kalinowski, A. Ragone, POLARIS: A framework to guide the development of Trustworthy AI systems, *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI (CAIN24), co-located with the 46th International Conference on Software Engineering, ICSE 2024*, 200-210, 2024

- [7] M. T. Baldassarre, **D. Gigante**, M. Kalinowski, A. Ragone, S. Tibidò, Trustworthy AI in practice: an analysis of practitioners' needs and challenges, *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering (EASE24)*, 293-302, 2024

- [8] M. T. Baldassarre, D. Caivano, B. F. Nieto, **D. Gigante**, A. Ragone, Fostering Human Rights in Responsible AI: A Systematic Review for Best Practices in Industry, *IEEE Transactions on Artificial Intelligence*, 1-15, 2024

Financial support

This research was financially supported by **SER&Practices** spin-off company

Contents

List of Figures

List of Tables

List of Abbreviations and Glossary

Abbreviations	
AI	Artificial Intelligence
ICT	Information and Communications Technology
KLOC	thousands (Kilo) of Lines Of Code
MbD	Model-based Design
PbD	Privacy by Design
PE	Privacy Engineering
PIA	Privacy Impact Assessment
PIRA	Privacy Impact and Risk Assessment
PKB	Privacy Knowledge Base
PMI	Project Management Institute
POSD	Privacy Oriented Software Development
PRE	Privacy Requirement Engineering
SDLC	Software Development Life Cycle
TAI	Trustworthy Artificial Intelligence

Glossary of Terms	
ChatGPT	ChatGPT (an acronym for Chat Generative Pre-trained Transformer) is a chatbot based on artificial intelligence and machine learning, developed by Open AI and specializing in conversation with a human user: it has shown remarkable capabilities in generating text similar to that used by people, so much so that it passed the Turing test with flying colors.
GenAI	Generative artificial intelligence (Generative AI, GenAI, or GAI) is a subset of artificial intelligence that uses generative models to produce text, images, videos, or other forms of data. These models learn the underlying patterns and structures of their training data and use them to produce new data based on the input, which often comes in the form of natural language prompts.

Fortify	Not intended as a static analysis tool for code quality in general, but specifically to analyze and test applications for security vulnerabilities, Fortify Static Code Analyzer (Fortify SCA), has been named for the 9th time one of the market leaders by Gartner in their application security testing market study.
PbD	Privacy by design aims to ensure the existence of the correct level of privacy and protection of personal data from the planning (design) stage of any system, service, product or process as well as during their life cycle. In other words, the principle of Privacy by design aims to ensure a proper level of data protection in all processing and implementation activities carried out within an organization.
PKB	It is the POSD Knowledge Base. PKB is made of the 5 Key Elements: Principles of Privacy by Design, Privacy Design Strategies, Privacy Patterns, Vulnerabilities and Context.
POLARIS	POLARIS is a framework designed to provide actionable guidelines and tools in order to support stakeholders in addressing TAI principles throughout the entire Software Development Life Cycle (SDLC).
POSD	It is a framework based on a knowledge base that links PbD, Privacy Design Strategies and Privacy Patterns concepts, across the entire SDLC, so that can be used by different kinds of stakeholders during the entire software development process.
SAST	Static application security testing (SAST) is used to secure software by reviewing the source code of the software to identify sources of vulnerabilities.
SDLC	In software engineering, is a process of planning and managing software development. It typically involves dividing software development work into smaller, parallel, or sequential steps or sub-processes to improve design and/or product management.
SonarQube	SonarQube is a SAST tool that analyzes code quality based on a customizable quality model and applies the concept of technical debt to inform developers not only about code quality issues but also about the estimated (accumulated) cost of removing those issues.
SonarCloud	SoftwareAsAService (SaaS) version of SonarQube.
TAI	Trustworthy AI refers to artificial intelligence systems designed and deployed to be transparent, robust and respectful of data privacy. Trustworthy AI makes use of a number of Privacy-enhancing technologies, including homomorphic encryption, federated learning, secure multi-party computation, differential privacy, zero-knowledge proof.

Chapter 1

Introduction

1.1 Motivations

Nowadays, organizations are undertaking increasingly complex projects in globalized, uncertain and dynamic environments. Some of the factors that increase or generate project complexity are the proliferation of multi-organizational international programs, the emergence of new technologies, the growing sophistication of projects' scope characterized by challenging technical, time and cost requirements and the increasing number of stakeholders involved [12].

Complexity influences project planning, coordination, and control; hinders the clear identification of goals and objectives of major projects; can affect the selection of an appropriate project organization form and experience requirements of management personnel; can be used as a criteria in the selection of a suitable project management arrangement; can affect different project outcomes such as time, cost, quality, security, etc. [13].

Moreover, the speed of the digitalisation process, also as a consequence of the just passed pandemic crisis, is leading to an exponential growth of threats to the privacy and security of software systems and a significant widening of the attack perimeter, due to the huge number of technologies used nowadays within a software project.

Another relevant trend is represented by the increase of "AI-enabled systems", i.e. software systems strongly based on Artificial Intelligence, e.g., autonomous, and semi-autonomous driving vehicles, in which vulnerabilities are not only of an applicative nature but may also arise from the inability of the algorithms used to correctly analyze atypical and unexpected behavior or, again, emerge from the combined use of software systems and physical devices (e.g. a sensor).

In these new complex and dynamic environments, project managers must rethink the traditional definition of a project and the ways to manage it [13] [14]; moreover, they must consider a new fundamental aspect: the trustworthiness of the system.

Trustworthiness is a complex and vague term, for which the literature has not yet found a unique and globally-accepted definition. A lot of concepts are involved when dealing with trustworthiness, privacy and security among them [15].

This imposes — see for example the provisions on the subject dictated by the "General Data Protection Regulation" (GDPR) [16] — the adoption of organizational structures, processes and tools to mitigate the risks related to security and data privacy. Other than the traditional vulnerabilities that can arise from the source code of a software system, in software development projects software errors can be introduced by disconnects and miscommunications during the planning development, testing and maintenance of the components [17]. Moreover, in AI-enabled systems, the vulnerabilities are not only of an applicative nature but may also arise from the inability of the algorithms used to correctly analyse atypical and unexpected behaviour or, again, emerge from the combined use of software systems and physical devices.

So, in order to satisfy security and privacy requirements, it becomes necessary to identify the threats, vulnerabilities and risks that may arise in the whole life cycle of a system (and not only in the development phase), in all the software components — even if of different nature — that compose the system; vulnerabilities could directly or indirectly damage (e.g. economic, political, social, reputation) an organization or individual and, in general, project stakeholders.

Therefore, starting from these considerations, the objective of this industrial research is to propose and experiment evolved software guidelines and tools capable of operating in both traditional and AI-enabled scenarios, for supporting both technical and non-technical stakeholders in the development of secure software systems. Specifically, the objectives to be achieved are:

- Goal 1: Proposal of guidelines and tools to support the development process of conventional software systems.
- Goal 2: Proposal of guidelines and tools to support the development process of AI-enabled systems in a company;
- Goal 3: Formularization of a theoretical framework which encopasses all above guidelines and tools and provides support for both technical and non-technical stakeholders who participate in a corporate project.

To achieve Goal 1, I conducted a literature review of the existent frameworks in the literature, selected the one with the lowest number of inefficiencies and then enhanced it with a number of experiments and field studies. To achieve Goal 2, as well I conducted a literature review of the existent frameworks in the literature. In this case, I found very few frameworks in the literature and no one fitted my research needs. For this reason, and to achieve Goal 3, I designed, implemented and evaluated a new framework specific to AI-enabled systems.

1.2 Main contribution

The main contributions of the work are summarized here:

- Contribution 1: Literature review of the existent frameworks that enable the security- and privacy-oriented development of traditional and AI-enabled software systems;
- Contribution 2: Empirical validation of the most comprehensive existent framework for security- and privacy-oriented development of traditional software systems;
- Contribution 3: Design, implementation and evaluation of a new framework for security- and privacy-oriented development of AI-enabled software systems, named POLARIS.

Here is a graphical scheme about how goals and contributions relate to each other:

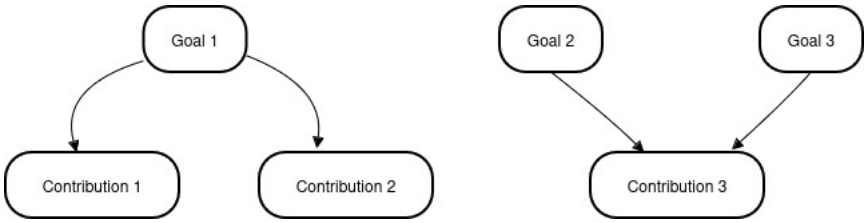


Figure 1.1: Goals vs Contributions relation.

1.3 Thesis Outline

The following thesis is organized as follows:

- **Chapter 1: Introduction**. The first chapter (current) gives an overview of the thesis, showing the motivations, research goals and contributions,
- **Chapter 2: Background and Preliminary Research**. This chapter encompasses some preliminary definitions needed to understand the context of this work, as well as the state of the literature prior to this work,
- **Chapter 3: Conventional Systems**. This chapter describes the author's research conducted in the conventional systems field and the resulting contribution to the literature,
- **Chapter 4: AI-based Systems**. This chapter describes the author's research conducted in the AI-based systems field and the resulting contribution to the literature,
- **Chapter 5: Conclusions and Future Works**. This chapter ends the thesis with an outline of conclusions, research results, and prospects for future works.

Chapter 2

Background and Preliminary Research

2.1 Famous software failures

A system failure could lead to an expensive and difficult resolving process, as well as bad customer reviews, unexpected costs for the fixing and competition loss. In order to avoid this situation, the use of error monitoring solutions has become very common in recent years. Anyway, a lot of companies still view monitoring as a costly optional extra, ignoring the fact that poor quality software costs a lot of money, e.g. US organizations spend over $2 trillion annually to fix faulty code[1].
There are a number of factors behind the financial cost of software errors. First of all, there is the direct cost of paying developers and software engineers to fix the problem in the source code. Secondly, there is the (eventual) cost associated with downtime and lost data (with its privacy consequences). Finally, there may be the cost of reputational damage - in case a lot of competitors exist. This happens because an organization in which a catastrophic software error happens may lose credibility with its customers and suppliers; in some cases, also violates the promised Service Level Agreement (SLA). All these factors end up in long-term financial losses. Anyway, there are not only monetary losses: there may also be negative outcomes related to users, such as human rights violations like privacy and even safety.
Some common reasons behind software failure are:

- Bad architecture definition and low-level design.
- Too strict schedule and/or milestone dates without enough background analysis.
- No incremental updates and adjustments for the requirements.
- Use excessive personnel to achieve unrealistic schedule compression.
- Miscommunication, egos and negative attitudes.

[1] `https://www.it-cisq.org/pdf/CPSQ-2020-report.pdf`

In the following paragraphs, there is a list of software incidents, which can help have an overview of causes and losses due to software failures.

The Mariner 1 Spacecraft - 1962

This failure is related to the NASA mission which launched a data-gathering unmanned spacecraft to fly past Venus, named "*The Mariner 1*".
What happened is that the Mariner 1 space probe just managed to leave Cape Canaveral; then, the rocket immediately veered dangerously off course. Since there was the risk of crash-landing on Earth, NASA engineers issued a self-destruct command. The craft self-destroyed about 290 seconds after launch.
A post-mortem investigation revealed the cause was a combination of two failures: an antenna hardware failure and an onboard guidance system software failure. The guidance antenna performed below its specification. So, the spacecraft had to rely on its onboard guidance system, which had a bug in it. Specifically, a programmer incorrectly transcribed a formula into computer code, missing a single hyphen, which was necessary to obtain the n-th smoothed value of the time derivative of a radius R. Without this smoothing function the software treated normal variation of velocity as if they were problematic, causing vague corrections that sent the spacecraft off course.
The overall cost of the omission has been $18.5 millions[2].

Hartford Coliseum Collapse - 1978

In 1978, some hours after thousands of visitors had left the Hartford Coliseum, the steel-latticed roof collapsed under the weight of wet snow.
This happened because the programmer of the CAD software - used to design the coliseum - incorrectly assumed the steel roof supports would only face pure compression. Also, the computer model assumed all of the top chords were laterally braced, but in fact only the interior frame met the criteria. As a consequence, dead loads were underestimated by more than 20%. When one of the supports unexpectedly buckled from the snow, it set off a chain reaction that brought down the other roof sections.
This cost $70 million of damage, plus another $20 million damage to the local economy.

Soviet Gas Pipeline Explosion - 1982

The Soviet gas pipeline explosion is considered to be an example of excellent cyber espionage, carried out by the CIA.
In 1982, during the Cold War between the USA and the USSR, the Soviet government built a gas pipeline based on advanced automated control software. Initially, the Soviets planned to steal this software from a Canadian company that specialized in this kind of software.
Anyway, the CIA was informed and began a counter-espionage action. CIA worked with

[2] `https://nasa.fandom.com/wiki/Mariner_1`

the Canadians to place intentional bugs in the software (also known as a *Trojan Horse*s) to compromise the Soviet pipeline.
The Soviets went ahead, stealing the compromised software and applying it to the pipeline. In June 1982, an explosion occurred with a force which was visible from space. Of course, this explosion irremediably damaged the pipeline, which cost tens of millions of dollars to construct and should have produced $8 billion in natural gas revenue.

Soviet missile detection system failure - 1983

The Soviet early warning system erroneously warned the United States had launched five ballistic missiles. The Soviet duty officer stated he had a "funny feeling in my gut" and reasoned if the U.S. were really attacking, they would have launched more than five missiles; so he manually reported the apparent attack as a false alarm.
This failure happened due to a bug in the Soviet software, which failed to filter out false missile detections because of sunlight reflecting off cloud-tops.
In this case, there were no economic or financial losses but human lives - in case of a third world war.

Medical Machine Kills - 1985

In 1985, Canada's Therac-25 radiation therapy machine failed and delivered lethal radiation doses to patients. The cause was concurrent programming errors (also known as *race conditions*), which sometimes gave the patients radiation doses that were hundreds of times greater than normal, resulting in death or serious injury. This happened due to two software errors. One emerged when the operator incorrectly selected the X-ray mode before quickly changing to electron mode, allowing the electron beam to be set for X-ray mode without the X-ray target being in place. The second fault allowed the electron beam to activate during the field-light mode, during which no beam scanner was active or target was in place.
Previous models had hardware interlocks to prevent such faults, but these were absent in Therac-25, which depended only on software checks for safety.
Totally, this cost radiation poisoning to six patients; in three cases, the injured patients later died as a result of the overdose.

Wall Street Crash - 1987

During the famous "Black Monday" - on October 19, 1987 - the Dow Jones Industrial Average plummeted 508 points, losing 22.6% of its total value. The S&P 500 dropped 20.4%. This was the greatest loss Wall Street ever suffered in a single day. The cause was particular. First of all, a long bull market was halted by a series of investigations made by the SEC on insider trading. So investors started selling their stocks: computer trading programs generated a flood of sell orders which resulted in a overwhelm of the market, crash of systems and immobilization of the investors.

The overall cost was $500 billion in just one day.

AT&T Lines Interruption - 1990

In 1990, a single switch in one of AT&T's 114 switching centers had a minor mechanical problem and resulted in shutting down the center. Once the centre came back up, it sent a message to other switching centres, which in turn caused them to shut down. This chain of events led to a down of the entire AT&T network for 9 hours.
The cause behind the fact that each centre, in turn, went off was a single line of buggy code in a complex software upgrade implemented to speed up callings.
The overall cost was 75 million phone calls missed and 200 thousand airline reservations lost.

Patriot Fails Soldiers - 1991

During the first Gulf War, an American Patriot Missile system in Saudi Arabia failed to intercept an incoming Iraqi Scud missile. On the contrary, the missile destroyed an American Army barracks. The problem was an inaccurate calculation of the time since boot, due to computer arithmetic errors. More in detail, the time was measured in tenths of a second by the system's internal clock; then, multiplied by 1/10 to return the time in seconds. This calculation was performed using 24 24-bit fixed point register. That means 1/10 value cut at 24 bits after the radix point. This led to a significant error, causing the missile to travel more than half a kilometre. So, the root cause is a rounding error.
This time, the overall cost was the death of 28 soldiers and the injury of 100 other soldiers.

The Morris Worm - 1988

This software bug was caused by a Cornell University student, who created a worm as part of an experiment. Anyway, the worm started replicating on all other machines and led them to crash because of a coding error. This is considered also the first computer virus since all the computers were connected through an early version of the internet. The student, Robert Tappan Morris, was flagged as a criminal hacker and fined $10,000[3], even if the cost of the damage was estimated to be between $10000 and $10 million[4].
This incident is now known because exposed a vulnerability and consequently improved the cyber-security posture. Nowadays, the worm's source code is kept as a museum piece on a floppy disk at the University of Boston.

Phobos 1 malfunction - 1988

On 2 September 1988, the expected transmission from Phobos 1 was not received. This was traced to a faulty key command that was sent on 28 August from ground control in Yevpatoria. A technician unintentionally left out a single hyphen in one of the keyed commands. All commands

[3] `https://www.fbi.gov/news/stories/morris-worm-30-years-since-first-major-attack-on-internet-110218`

[4] `https://www.captechu.edu/blog/cyber-security-impact-30th-anniversary-of-morris-worm`

were supposed to be validated by a computer before being transmitted, but the computer that checked the code was malfunctioning. The technician violated the internal procedure policies and transmitted the command before the computer could be fixed to proofread it. This minor alteration in the code deactivated the attitude thrusters. By losing its lock on the Sun, the spacecraft could no longer properly orient its solar arrays, thus depleting its batteries.
Moreover, the software instructions used to turn off the probe's attitude control - normally considered a fatal operation - were part of a routine used when testing the spacecraft on the ground. Ideally, this routine should not be present before launch. However, the software was coded in PROMs, and so removing the test code would have required removing and replacing the entire computer. Because of time pressure from the impending launch, engineers decided to leave the command sequence in, though it should never be used. However, a single-character error in constructing an upload sequence resulted in the command executing, with subsequent loss of the spacecraft.
Phobos 1 was launched July 7 and Phobos 2, its successor, five days later in missions that cost $480 million[5].

London Ambulance - 1992

In 1992 the London Ambulance Services (LAS) decided to replace human operators with a computer system named CAD. It was put into operation without load testing and with 81 known bugs. The system crashed due to errors in the distribution of routes of ambulances, which resulted in the death of a number of people. Furthermore, the equipment was cheap and broke down a couple of hours after active use of the system.
The story behind this system is complex. After a number of logistics issues, including a project cancellation and re-design, the software system was developed and deployed on the morning of October 26, 1992. Just a few hours later, however, problems emerged. The visible effect was the inability to keep track of the ambulances and their statuses in the system. It began sending multiple units to some locations and no units to other locations; the efficiency with which it assigned vehicles to call locations was substandard. The system began to generate such a great quantity of exception messages on the dispatchers' terminals that calls got lost. Moreover, the overall traffic increased because people called back additional times since the ambulances they were expecting did not arrive. As more and more incidents were entered into the system, the system reached saturation. The next day, LAS switched back to a part-manual system. Eight days later the system was completely shut down and abandoned.
Software system problems, in this story, are many: first of all, due to bad data validation, the software system did not function well when given invalid or incomplete data regarding the positions and statuses of ambulances.
Secondly, the system had a number of problems in different parts of the user interface. For example, parts of the terminal screens had black spots, thus preventing ambulance operators

[5] `https://apnews.com/article/8e65a6dc038aa841808008a62593a9f3`

from getting all the information needed. When ambulance crews attempted to remedy mistakes after pushing incorrect buttons, the system did not accept their fix input. Again, the software failed to prevent and mitigate the error conditions that occur in normal, day-to-day operations. However, the real main problem in the system was a memory leak in a small portion of the code. This defect retained memory that held incident information on the file server even after it was no longer needed. As with any memory leak, after enough time, the memory filled up and caused the system to fail.
Because of the large area serviced by LAS, a lot of people were directly affected by the computer system failure. There were as many as 46 deaths that would have been avoided if the requested ambulance arrived on time. One heart attack patient waited six hours for an ambulance before her son took her to the hospital. Four hours after that, LAS called to see if the ambulance was still needed. Another woman called LAS every thirty minutes for almost three hours before an ambulance arrived. It was too late, as her husband had already died. One ambulance crew arrived only to find that the patient had not only died but his body had been taken away by a mortician.

Pentium FDIV Bug - 1994

The story behind this bug started when Thomas Nicely, a math professor, discovered a flaw in the Pentium processor. He then reported it to Intel and their response was they would have offered a replacement chip to anyone who could prove he was affected by it.
The problem was a hardware bug affecting the floating-point unit (FPU) of the P5/Pentium processors. Everything started from missing values in a lookup table used by the FPU's floating-point division algorithm, which resulted in small errors in calculations. The outcome was some inaccuracies in very specific calculations, which occur very rarely. In fact, the chance of a miscalculation was estimated to be just 1 in 360 billion[6]. For example, dividing 4195835.0/3145727.0 yielded 1.33374 instead of 1.33382, an error of 0.006%.
Even if the concrete effects of the software error were negligible, millions of people requested a new chip - costing Intel around $475 million - because the details of the bug were publicly disclosed by the international press.

ESA Ariane 5 Flight V88 - 1996

The European Space Agency's Ariane 5 failure was caused by more than one error.
Just 36 seconds after its launch, the rocket engines failed due to software errors. The Ariane 5 reused the code from Ariane 4, but the early part of the Ariane 5's flight path differed from the Ariane 4 in having higher horizontal velocity values. As a consequence, the internal value BH (Horizontal Bias) calculated in the alignment function was unexpectedly high. Anyway, the alignment function was operative for approximately 40 seconds of flight, which was based on a requirement valid also for Ariane 4. After the lift-off, this function served no purpose on the Ariane 5. These greater values of BH caused a data conversion from a 64-bit floating

[6] `https://www.cs.earlham.edu/~dusko/cs63/fdiv.html`

point number to a 16-bit signed integer value due to overflow and raised a hardware exception. The programmers had protected only four out of seven critical variables against overflow due to problems with the computational constraints of the on-board computer. So, they relied on assumptions about the possible range of values for the three unprotected variables. This was correct for the Ariane 4 - but not the Ariane 5 - trajectory. The raised exception halted both the inertial reference system modules, although they were intended to be redundant. Then, the active module presented a diagnostic bit pattern to the onboard computer which was interpreted as flight data. This led to an angle of attack of more than 20 degrees, causing separation of the boosters from the main stage, the triggering of the self-destruct system of the launcher, and the destruction of the flight.
The failure resulted in a $370 million loss for the ESA, but a lot of recommendations were issued after the subsequent investigation, including the need for improved software analysis and evaluation.

Mars Pathfinder - 1997

After a few days into the mission named "Mars Pathfinder", while Pathfinder started gathering meteorological data, the spacecraft began experiencing total system resets, resulting in losses of data.
Later, in an IEEE Real-Time Systems Symposium keynote, David Wilner - Chief Technical Officer of Wind River System, which makes VxWorks, the real-time embedded systems kernel that was used in the Mars Pathfinder mission - explained in detail the actual software problems that caused the total system resets, how they were diagnosed, and how they were solved.
VxWorks provided preemptive priority scheduling of threads. Tasks on the Pathfinder spacecraft were executed as threads with priorities that were assigned in the usual manner reflecting the relative urgency of these tasks.
Pathfinder contained an "information bus", a shared memory area used for passing information between different components of the spacecraft. A bus management task ran frequently with high priority to move certain kinds of data in and out of the information bus. Access to the bus was synchronized with mutual exclusion locks (technically known as "*mutexes*").
The meteorological data gathering task ran as an infrequent, low-priority thread, and used the information bus to publish its data. When publishing its data, it would have acquired a mutex, done writes to the bus, and released the mutex. If an interrupt caused the information bus thread to be scheduled while this mutex was held, and if the information bus thread then attempted to acquire this same mutex in order to retrieve published data, this would cause it to block on the mutex, waiting until the meteorological thread released the mutex before it could continue. The spacecraft also contained a communications task that ran with medium priority. Most of the time this combination worked fine. However, very infrequently it was possible for an interrupt to occur, causing the (medium priority) communications task to be scheduled during the short interval while the (high priority) information bus thread was blocked waiting

for the (low priority) meteorological data thread. In this case, the long-running communications task, having higher priority than the meteorological task, would have prevented it from running, consequently preventing the blocked information bus task from running. After some time had passed, a watchdog timer would have gone off, noticed that the data bus task had not been executed for some time, concluded that something had gone drastically wrong, and initiated a total system reset.
This scenario represents a classic case of priority inversion.
David later also stated that one or two system resets had occurred some months before the flight, during the test phase. They had never been reproducible or explainable and so the engineers, in a very human-nature response of denial, decided that they probably were not important, using the rationale "it was probably caused by a hardware glitch". Moreover, engineers were extremely focused on ensuring the quality and flawless operation of the landing software. Should it have failed, the mission would have been lost. So, the engineers ignored occasional glitches in the less-critical land-mission software, particularly given that a spacecraft reset was a viable recovery strategy at that phase of the mission.
Due to this failure, NASA lost its $125-million aircraft[7].
Again, thinking and running a test suite with edge cases would have helped identify these strange and low-likely cases.

USS Yorktown Incident - 1997

On 21 September 1997, the USS Yorktown halted for almost three hours due to a divide-by-zero error in a database application that propagated throughout the ship's control systems.
The Yorktown had successfully served the US Navy since 1984 without a major incident during multiple combat operations; however, as part of an IT modernization program dubbed "Smart Ship", its control systems were modified in 1996 to use a network of PCs running Windows NT 4.0.
The problem was raised when the USS Yorktown was performing training exercises off the coast of Cape Charles, Virginia. A crew member began troubleshooting a fuel valve that was physically closed, but according to the Smart Ship's Standard Machinery Control System (SMCS) was open. The technician tried to digitally calibrate and reset the fuel valve and entered a 0 value for one of the valve's component properties into the SMCS Remote Database Manager (RDM). The RDM program then attempted to perform a division operation by the valve property; a *divide-by-zero arithmetic exception* was thrown, not caught by the program, and the RDM crashed. Since other Smart Ship systems were dependent on RDM availability across the LAN, these other SMCS components - including ones controlling the motor and propulsion machinery - began to fail in a domino-like sequence until the ship stopped in the water. The crew was able to troubleshoot and restart the ship's systems after two hours and forty-five minutes, and the Yorktown returned to base in Norfolk, Virginia.

[7] `https://www.latimes.com/archives/la-xpm-1999-oct-01-mn-17288-story.html`

Even if no human disasters happened, again the problem could have been avoided by carefully testing the code with edge integer values and catching the possible exceptions.

British Passports to Nowhere - 1999

In 1999, the U.K. Passport Agency acquired and started using a new Siemens computer system, which cost £230 million. This automated system failed to issue passports on time for more than a half million British citizens. To compensate the disaster, the Agency had to pay £12.6 million: £6m for staff overtime, £16,000 for umbrellas for applicants forced to queue all day in the rain for their passports and £161,000 compensation to holidaymakers.
The root cause was that the Passport Agency introduced its new system without adequately testing it. Furthermore, the staff was not adequately trained to use it. At the same time, there was a law change which imposed all children under 16 travelling abroad to have a passport, resulting in a huge amount of passport demands, which overwhelmed the (buggy) new computer system.

Radiation therapy software by Multidata Systems International - 2000

As in most radiotherapy departments, the one at ION uses a treatment planning system (TPS) to calculate the resulting dose distributions and determine treatment times. Each shielding block data should be entered separately into the TPS. The TPS by Multidata Systems International allowed a maximum of four shielding blocks per field to be taken into account when calculating treatment times and dose distributions. Just to clarify, shielding blocks are generally used to protect healthy tissue of patients undergoing radiotherapy at the Institute.
A radiation oncologist requested to include five blocks in the field. In order to satisfy this request, in August 2000 the method of digitizing shielding blocks was changed.
A post-mortem analysis discovered that it was possible to enter data into the TPS for multiple shielding blocks together as if they were a single block, thus apparently overcoming the limitation of four blocks per field. Although the TPS accepted the entry of the data for multiple shielding blocks as if they were a single block, was also found that one of the ways in which the data was entered in the computer resulted in a treatment time longer than it should have been. As a consequence, patients received a higher dose than that prescribed. The modified treatment protocol was used for 28 patients, treated between August 2000 and March 2001 for prostate cancer and cancer of the cervix. 8 died and 20 got injuries.
The cause of this disaster was that the modified protocol was used without a verification test. In other words, no manual calculation of the treatment time for comparison with the computer-calculated treatment time was done, nor was a simulation of treatment by irradiating a water phantom and measuring the dose delivered. That is why the error was not identified earlier.

The Millennium Bug - 2000

The Millennium Bug, or Y2K, was a massive concern raised in 1999. The problem was that

computer systems would not be able to deal with dates after December 31, 1999, since most computers and operating systems only used two digits to represent the year, ignoring the 19 prefix.
Even if this did not cause too many real-life problems - because most systems made adjustments in advance -, the fear cost thousands of considerable amounts of money in rigorous planning and preparations for institutions, businesses and even families. As an example, was estimated that the USA totally spent $100 billion to prevent the bug.

NASA's Spirit rover - 2004

A shortage of memory on board the Spirit Mars rover caused it to become unresponsive on the Martian surface on 22 January 2004.
Mike Deliman, a technical staff member at Wind River Systems Inc., the company which provided the real-time embedded operating system used in the mission, stated the problem appeared to be entirely memory-related: "It's not a software bug, it's not an application bug, and it's not a hardware bug," Deliman said. "It's a system constraint that we ran up against."
The constraint in Spirit Rover is 32MB of its 128MB of RAM are dedicated to the onboard Wind River VxWorks operating system and some other applications. As the mission progresses, technicians are required to periodically delete old files and directories to clear out the memory for reuse.
As one could easily expect, this step was not performed quickly enough by mission technicians. As a consequence, the rover just ran out of RAM memory resulting in a connection loss with the ground station.
Technicians were able to correct the problem when the rover went into diagnostic mode, Deliman said. By sending diagnostic commands, the technicians deleted a series of files and folders from a flash-memory-based file system board. So the rover resumed normal operations.
In this case, there were no monetary or human lives lost.

EDS Child Support System - 2004

In 2004, the UK government started utilizing a system to support the operations of the Child Support Agency (CSA). The supplier was the IT company Electronic Data Systems (EDS) and the provided system was called CS2.
An internal report was leaked and this revealed that the system was "badly designed, badly tested and badly implemented". Later, CSA reported that CS2 "had over 1000 reported problems, of which 400 had no known workaround", resulting in "around 3000 IT incidents per week". Numerically speaking, the system accidentally overpaid 1.9 million people, underpaid another 700,000, had £3.5 billion in uncollected child support payments, a backlog of 239,000 cases and 36,000 new cases "stuck" in the system.
The system was budgeted to cost around £450 million but ended up costing £768 million altogether. Moreover, EDS declared a $153 million loss in their subsequent financial results.

Air-Traffic Control System in LA Airport - 2004

In 2004, the controllers lost contact with the planes when the main voice communications system shut down unexpectedly. Moreover, a backup system that was supposed to come up in such an event crashed a minute after it was turned on. The outage disrupted about 800 flights across the country.

The Palmdale system that shut down, causing all the chaos, was a Voice Switching and Control System (VSCS), one of 21 in use throughout the continental United States and Alaska. Designed by Harris Corp., Melbourne, Fla., it has been running in air traffic control facilities since the mid-1990s. With the VSCS, controllers used a touch-screen to select a phone line to connect them to other controllers or to a radio frequency to talk to flight crews. It was a complex system, according to Richard Riggs, a spokesperson for the Professional Airways Systems Specialists, the union of technicians who maintained the communications systems for the FAA.

On 14 September, the FAA stated the problem was caused by human error: "Our preliminary findings indicate that the outage was not the result of system reliability but rather an event that should've been avoided if strict FAA operating and maintenance procedures had been followed." These procedures required that a technician reboot the voice switching system every 30 days. Anyway, this reboot was necessary due to a software glitch, Riggs then stated. The glitch resided in an auxiliary system: the VSCS Control Subsystem Upgrade (VCSU), developed by Harris as well. The VCSU was the control system for the VSCS and checked its health by continually running built-in tests on the system. It was also used when loading new data and software into the VSCS. It contained a countdown timer that ticked off time in milliseconds; the VCSU used this timer as a pulse to send out periodic queries to the VSCS. It started out at the highest supported number: 2 elevated to 32, just over 4 billion milliseconds. When the counter reached zero, the system ran out of ticks and shut down. Counting down from 2 elevated to 32 to zero in milliseconds takes just under 50 days. The FAA procedure of having a technician reboot the VSCS every 30 days resulted in resetting the timer to 2 and elevating it to 32 almost three weeks before it ran out of digits. Jim Turley, an independent embedded-processor analyst, said many computing systems had such type of timer. The correct procedure should be that the software automatically reloads or the timer automatically resets itself before the timer expires. Luckily, no disasters happened but Hamid Ghaffari, president of a NATCA local, stated: "Had this happened 10 or 15 years ago, when there was no onboard collision avoidance system, you would have had several midair collisions".

Microsoft Zune's New Year Crash - 2008

This flaw is related to a bug in the software code produced by Freescale, the semiconductor company spun off from Motorola.

The Zune's real-time clock stored the time in terms of days and seconds since January 1st, 1980. When the Zune's clock was accessed, the driver displayed the number of days into

years/months/days and the number of seconds into hours/minutes/seconds. Likewise, when the clock was set, the driver did the opposite.
The error was simple: in the source code, there was a loop to handle leap years, like 2008 was. Anyway, a broken loop blocked the algorithm to get past the beginning of the 366th day of the year.
In 2008, the algorithm fell into a bad loop on the last day: the number of days was greater than 365 and it was a leap year. So the device ran around the loop forever - or at least until the real-time clock could show that it was 2009 (so 24 hours later).
This problem demonstrates again the importance of testing edge conditions.
Although customers were alerted this was not a permanent malfunction, some of them lost loyalty in the company. Of course, a number of Zune owners wrote on online forums and blogs with their testaments of ire and frustration, some threatening to switch to Apple's iPod if the issue was not remedied.
Microsoft said that it has sold 3 million Zunes since introducing the device in November 2006, but that this bug affected only the unspecified number of models with 30 gigabytes of memory that were manufactured by Toshiba Corp. and later discontinued in 2008.

Bitcoin Hack, "Mt. Gox" - 2011

Mt. Gox was the biggest and most used bitcoin exchange in the world in 2010; once a software bug emerged, it closed. The bug led to the creation of wrong transactions that could never be fully redeemed; this cost around $1.5 million bitcoins, which went lost. This originated from a problem in the script ran for every transaction, which checked that the public key in the transaction matched the receiver address, and the signature was valid, proving the spender of the bitcoins owned the private key; if everything matched, the bitcoins could be spent. To trigger the bug, was sufficient to put a 0 as the receiver address: this was interpreted as OP_0 by the algorithm, resulting in pushing an empty array of bytes as the receiver address. Since it was calculated a 160-bit hash of the receiver address, and it was impossible that this matched an empty array, the script ended with an error and the related bitcoins could never be spent.
Later, in 2014, the exchange lost more than 850,000 bitcoins (valued at around half a billion USD at the time) in a hacking incident. Even if around 200,000 bitcoins were recovered, the exchange ended up declaring bankruptcy.
This second incident was declared to be caused by the *transaction malleability* of the transactions. This property describes the fact that the signatures that prove the ownership of bitcoins being transferred in a transaction do not provide any integrity guarantee for the signatures themselves. So, an attacker could mount a malleability attack in which it intercepts, modifies, and rebroadcasts a transaction, causing the transaction issuer to believe that the original transaction was not confirmed. Anyway, Decker et al. [18] used traces of the Bitcoin network for over a year preceding the incident and showed that there was no widespread use of malleability attacks before the closure of MtGox.

Knight's $440M in bad trades - 2012

In 2012, Knight Capital Group invested in a new trading software that was supposed to help them gain a huge advantage in the stock markets. On the contrary, it caused the bankruptcy of the firm. This was caused by several software errors combined, which led Knight to make a series of purchases and spend more than $7 billion on 150 different stocks.
In just 30 minutes, 75% of the Knight's value disappeared. The unintended trades cost $440 million and Goldman Sachs had to step in to rescue them. Anyway, Knight never really recovered and was acquired by a competitor less than a year later.

Boeing 787 Dreamliner - 2015

A software vulnerability discovered Boeing's new 787 Dreamliner jet had the potential to cause pilots to lose control of the aircraft, even in mid-flight. This was discovered by Boeing itself in 2015 before any disaster happened.
The bug belongs to the category of *integer overflow* and resided in one of the electrical systems responsible for generating power, according to what was reported in the FAA report.
The problem emerged when the aeroplane was powered continuously for 248 days, resulting in loss of all alternating current (AC) electrical because all the generator control units (GCUs) simultaneously went into failsafe mode. This condition was caused by a software counter, internal to the GCUs, that encountered an overflow after 248 days of continuous power. The plane had four main GCUs associated with the engine-mounted generators but, if all of them were powered up at the same time, after 248 days of continuous power all four GCUs would have gone into failsafe mode at the same time, resulting in a loss of all AC electrical power regardless of flight phase.
Some informed speculation later stated it was a signed 32-bit integer overflow that was triggered after 2 exposed to 31 centiseconds (i.e. 248.55 days) of continuous operation.

IPhone Unicode of Death - 2015

In 2015 a (not) new iOS bug emerged. This bug enabled a malicious actor owning an iPhone to crash a victim's iPhone by simply sending a text message. The text message, visible in Fig. 2.1, caused the iPhone of the recipient to crash continuously if the text was received while in lock screen mode. The "Effective Power" bug (also known as "Unicode of Death") could be triggered and cause issues only between iPhone-to-iPhone communication.

OpenSSH RegreSSHion - 2024

On July 1, 2024, a significant security vulnerability was discovered in OpenSSH, specifically impacting *glibc-based* Linux systems. This flaw, identified as CVE-2024-6387, poses a critical risk as it allows for unauthenticated remote code execution (RCE) with root privileges. The vulnerability stems from a signal handler race condition within the OpenSSH server component

Figure 2.1: Text message to trigger iPhone "Effective power" bug.

(sshd), which is responsible for accepting connections from client applications. An in-depth security analysis has revealed that this flaw is essentially a regression of the previously patched CVE-2006-5051 vulnerability, which was addressed 18 years ago. Unfortunately, it was inadvertently reintroduced with the release of OpenSSH version 8.5p1 in October 2020. This recent discovery serves as a stark reminder of the critical importance of comprehensive regression testing to prevent the reoccurrence of previously resolved vulnerabilities.

CrowdStrike Faulty Update - 2024

On 19 July 2024, American cybersecurity company CrowdStrike distributed a faulty update to its Falcon Sensor security software that caused widespread problems with Microsoft Windows computers running the software. As a result, roughly 8.5 million systems crashed and could not properly restart in what has been called the largest outage in the history of information technology and "historic in scale". The outage disrupted daily life, businesses, and governments around the world. Many industries were affected—airlines, airports, banks, hotels, hospitals, manufacturing, stock markets, broadcasting, gas stations, retail stores, and more—as were governmental services, such as emergency services and websites. The worldwide financial damage has been estimated to be at least US$10 billion. Within hours, the error was discovered and a fix was released, but because many affected computers had to be fixed manually, outages continued to linger on many services.

Apple immediately published an official support document with instructions about how to temporarily fix the bug, while working to provide an official fix.

The crash was caused by the way Unicode decoded characters, which led to an overloading of the device's memory. This way, the victim's iPhone was continuously rebooted. Some users reported that they were not able to open the Messages app again after receiving the "Effective Power" message, according to iDownloadblog[8].

A similar Unicode bug was noticed in August 2013 when iOS 6 and OS X 10.8 were released. Apple's engineers patched that Unicode bug through the release of iOS 7 and OS X 10.9.

Also in this case, no human lives were endangered but many people on the web expressed their

[8] http://www.idownloadblog.com/2015/05/28/apple-publishes-temporary-fix-for-messages-issue-says-software-update-coming-soon/

unhappiness and some decided to abandon Apple, with a subsequent economic loss for the company.
There are also a lot of incidents here not reported because additional technical details were not provided by the manufacturers and/or the bug finders.

Hereon a list of incidents relating to AI-enabled software systems is presented, which demonstrates the urgency of investigating the safety and security aspects of these systems as well before their deployment in a real-world scenario [19]:

- In 2016, the Auto-pilot system in a Tesla car confused the white side of a truck with the sky and this caused a deadly crash[9].

- In 2016, the chatbot "Tay" developed by Microsoft was shut down and closed a few hours after its release time: the underlying model was attacked and forced to post offensive tweets against users[10]. This is very dangerous because chatbots are used not only in corporate' businesses but also for many government services. These smart systems are vulnerable to hacking as they consume data from common users (e.g: citizens) to analyze and manage their requests.

- In 2016, a Google Autonomous Vehicle (AV) encountered a failure in the speed estimation module and this caused a crash with a bus[11]. Luckily there were no injuries.

- In 2017, Apple's face recognition system was fooled using a cheap 3D-printed mask[12], allowing an attacker to unlock any iPhone exploiting the Face ID lock feature. Anyway, some technical details are still required to effectively state the presented PoC is valid.

- In 2018, Uber's self-driving cars killed a pedestrian because the auto-breaking system failed and not stopped at the right time[13]. Some investigators said the vehicle decided it needed to brake 1.3 seconds before striking the pedestrian, but Uber had previously disabled the Volvo's automatic emergency braking system in order to prevent driving interference. In this case, it is very important to note that failures may originate from the interaction between (intelligent) software and hardware.

- In 2018, Eykholt et al. [20] demonstrated that robust physical perturbations could fool the DNN-based classifier of a self-driving car to misclassify speed limit signs.

- In 2018, Carlini et al. [21] used targeted audio adversarial examples to successfully attack DeepSpeech, a deep learning-based speech recognition system; the main idea behind this

[9] https://www.tesla.com/en_JO/blog/tragic-loss
[10] https://blogs.microsoft.com/blog/2016/03/25/learning-tays-introduction/
[11] https://www.bbc.com/news/technology-35692845
[12] https://www.forbes.com/sites/thomasbrewster/2017/11/13/apple-face-id-hacked-by-vietnamese-researchers-mask/
[13] https://www.theverge.com/2018/5/24/17388696/uber-self-driving-crash-ntsb-report

work is that the underlying model works with sound waves which can also carry secret commands to the connected devices. So doing, they succeeded in sending executable commands to a speech-recognition-based device (like Amazon Alexa).

- In 2019, Tencent's Keen Security Lab successfully exploited the Tesla's autopilot system by performing small changes on the lane markings (still visible to human eyes): as a result, a Tesla Model S moved to the wrong lane, making the lane recognition models in Tesla risky and unreliable under some conditions [22].

- In 2019, a neural network model failed a diagnosis [23]: a benign mole was diagnosed as malignant because of little graphical noise in the original medical image.

- In 2019, a case of biased algorithm became famous: the smart algorithm guiding care for tens of millions of people was found biased against dark-skinned patients in New Jersey, USA; the effects were it was assigning dark-skinned patients lower scores than white patients with the same medical conditions[14]. This could cause wrong medical prescriptions.

- In 2020, the shopping guide robot employed in Fuzhou Zhongfang Marlboro Mall, China, started walking to the escalator by itself and subsequently fell off, knocking over some human customers[15]. Nowadays, no details about the possible causes have been released.

- In 2021, the National Highway Traffic Safety Administration revealed there were 273 crashes involving Tesla cars due to the autopilot feature, nearly 70 per cent of the 392 crashes involving advanced driver-assistance systems reported since the previous year[16]. In the crashes vehicles most frequently collided with fixed objects or other cars. When the vehicles reported damage, it was most commonly to the front of the car, which was the case in 124 incidents. Damage was more often concentrated on the front left, or driver's side, of the car, rather than the passenger's side.

- In 2021, FBI issued a warning about a dangerous increment in AI-based synthetic materials, including deepfake content, which can be used to spread fake news[17]. Since users are already duped by phishing attacks, deepfake phishing attempts would be even more difficult for the average user to detect. They also suggest cybersecurity awareness training is a must and should be updated to make sure it also includes how to tell a fake from the real deal.

- In 2021, the "Lee Luda" AI chatbot has been suspended (and the firm is now being sued in South Korea) for making homophobic slurs and leaking users information[18]. Here, again,

[14] https://www.nature.com/articles/d41586-019-03228-6
[15] https://incidentdatabase.ai/cite/134?lang=en
[16] https://www.washingtonpost.com/technology/2022/06/15/tesla-autopilot-crashes/
[17] https://securityintelligence.com/articles/how-protect-against-deepfake-attacks-extortion/
[18] https://www.ic3.gov/Media/News/2021/210310-2.pdf

the problem is Luda became a target by manipulative users as well, with online community boards posting advice on how to engage it in conversations about sex, including one that read: "How to make Luda a sex slave," along with screen captures of conversations.

Of course, this is a non-exhaustive list: a lot more incidents can be found in "AI Incident Database"[19].
This subset of incidents demonstrates that there are a number of issues to be handled beyond building an AI model with good performance.
Anyway, it must be noted some good-willing initiatives are also emerging. For example, in 2019 Facebook (now Meta) created a dataset and a challenge for deepfakes detection[20].

2.2 Global privacy landscape

Nowadays we are witnessing the biggest change in the evolution of digital technology, with the world transforming from a paper-based system to a fully digital one.

Many digital technologies are characterized by a steady increase in their use within the world thanks to the Internet, which now connects not only digital processors but also a wide variety of intelligent and fully automated devices that are now part of our daily lives, including electronic chips embedded in non-digital objects (e.g. RFID tags), sensors, actuators, and the various wearable devices [24]. The dependence towards information technology shown by the business world as well as many other sectors has significantly increased in the current era.

Various public and private sector organizations are becoming extremely dependent on information technology and its implementations because of the huge benefits they can reap in a very short time [25]. Web applications[21], as well as numerous other software, are based on Information Technology (IT)-enabled services, implemented through computers and networks with the ability to store, manage, transfer, and update data in data repositories sometimes belonging to different entities.

However, on the other side of the coin, we can see that the volume of cyber-attacks is increasing exponentially and, consequently, the demand to protect systems from external threats.

Nowadays more than ever, another key concept to take care of is *privacy* and *data protection*.

The terms privacy and data protection are often used interchangeably [26], but they are two separate concepts. Privacy, from the perspective of the individual, is the act of resisting the improper control of others. In contrast, data protection is a concept that exists from the perspective of the organisation, where laws and regulations prevent organisations from abusing their power over user data [27]. Solove [28] attempted to conceptualise privacy as the right to

[19] https://incidentdatabase.ai/cite/134?lang=en

[20] `https://ai.facebook.com/blog/deepfake-detection-challenge/`

[21] A web application (web app for short), in computer science and particularly in web programming, generically denotes all distributed applications, i.e., applications that can be accessed/used via the web by means of a network, such as an Intranet within a computer system or through the Internet.

be left alone in solitude, limiting access to the self, secrecy and control of access to personal information. In subsequent research, Solove expanded and developed a taxonomy [29] for privacy that includes four main groups of activities, information gathering, processing, dissemination, and invasion. Anyway, we are far from a privacy-friendly world: an alarming trend is that individuals rarely win lawsuits against private parties for privacy violations because privacy like freedom means "*so many different things to so many different people that it has lost any precise legal connotation that it might once have had*" [29, 30].

Privacy, in security terms, means that we are free to carry on a conversation with a fair amount of confidence that we are not leaking personal or business secrets to an unwanted third party. It is about choosing those we trust. UK Information Commissioners Office (ICO) [31] categorises privacy into four sections: (1) privacy of personal information, also known as data privacy and information privacy (2) privacy of the person, also known as bodily privacy (3) privacy of personal behaviour (4) privacy of personal communication.

Identifying and modelling privacy requirements during the preliminary stages of software application development is essential in order to provide a high degree of privacy protection for stakeholders, including users. Keeping software privacy safe, as well as maintaining data security, continues to be a challenge for software professionals as information systems become more complex and distributed. Implementing such software not only poses technical hurdles but also requires compliance with privacy laws. For example, privacy for healthcare applications has significant implications because service delivery affects human life, reputation, and government policies. The Institute of Medicine (IOM) study published in this context confirms this [32].

In this context, it is clear that preserving the privacy of the data used by the software is one of the focal points of the development of reliable and quality software systems. Therefore, it is necessary to consider the security and privacy of users at all stages of the software application lifecycle [33–36]. Several strategies focus on security requirements while very few mechanisms address privacy at design time [37]. Privacy and security requirements are factors that must already be considered when developing software or eliciting system requirements; this is especially important in heavily regulated domains and in the critical infrastructure sectors such as financial services, healthcare, energy, and others. Enforcing privacy requires an organization to focus on gathering requirements while also relying on what is set forth in laws or regulations.

A context in which privacy is a fundamental requirement is Healthcare. Here, the use of Information and Communications Technology (ICT)[22] is a growing need, given the many benefits it brings. These benefits can be observed in the improvement of patient care, such as reduction of medical errors, knowledge sharing among health care professionals, improvement of usability, accessibility, and consistency of information, as well as improving the overall quality of services.

Anyway, the processing and distribution of electronic health records (EHRs) has raised

[22]Information and communication technology is the set of methods and techniques used in the transmission, reception, and processing of data and information (including digital technologies)

privacy concerns for both patients and health agencies [38]. Lack of sufficient security in maintaining confidentiality, integrity, and accessibility of data leads to possible threats in this domain. Weak security procedures and little awareness of risk management practices are items to pay attention to. For this reason, industry best practices and standards in healthcare, e.g., *ISO/IEC 27002 (ISO 27799: 2008), Health Insurance Portability and Accountability Act (HIPAA), Patient Safety and Quality Improvement Act 2005 (PSQIA), Health Information Technology for Economic and Clinical Health (HITECH), Personal Information Protection and Electronic Documents Act (PIPEDA), General Data Protection Regulation GDPR*[23]*, Data Protection Act (DPA), IT Act* and many other pieces of legislation, have become necessary to protect the information assets within computers.

Many other regulations exist because different nations have different data protection policies and legislation. Some countries' data protection guidelines and laws are shown in Fig. 2.2.

Figure 2.2: Privacy laws/regulations of some countries.

Developing a regulatory system that can give guidance on technical developments requires that we look at the problem from a broader perspective. Specific privacy regulations and standards still depend heavily on the policies of each government. However, these laws have not been considered effective enough by experts and practitioners in the field. Many are aware of the discrepancy between the rate of technological growth and the management of that transformation through legislative mechanisms [40]. Danezis et al. [41] observed that, in general, when implementing desired functional features, privacy and data protection are neglected by conventionally used engineering methods. This neglect is caused and encouraged by limitations in developers' ability to understand, as well as lack of resources in pursuing privacy-by-design. As a result, there is a need to bridge the gap between the legislative framework for privacy and related implementation techniques [41].

[23]The European Union's GDPR implemented and replaced the *Protection Directive* as of May 25, 2018 [39]. It aims to improve the privacy protection of the individuals to whom the data relates.

2.3 IT security landscape

While in the previous section it was presented the global landscape regarding privacy regulations, here are presented the standards that focus on the *information security management* — that encompasses many areas, from perimeter protection and encryption to application security and disaster recovery. Implementing information security management measures makes it possible to protect the users' data, demonstrating an effort to guarantee their privacy.

IT security is made more challenging by the huge number of privacy regulations and standards, such as the ones presented in the previous section.

This is where IT *security frameworks and standards* come to the aid. Knowledge of regulations, standards and frameworks is essential for all cybersecurity professionals. On the other hand, compliance with these frameworks and standards is important from an audit perspective.

Standards list out steps to perform, and IT organizations must comply with requirements set forth in a standard.

Regulations, conversely, have a legal binding impact. The way they describe how to do something indicates government and public support for the rules and processes set forth in the regulation. Failure to comply with IT-focused regulations can result in financial penalties and litigation.

An *IT security framework* is a series of documented processes that define policies and procedures around the implementation and ongoing management of information security controls. These frameworks are a blueprint for managing risk and reducing vulnerabilities.

Frameworks are of vital importance because provide a starting point for establishing processes, policies and administrative activities for information security management. Using a common framework, an organization can establish crosswalks to demonstrate compliance with multiple regulations

Information security professionals use frameworks to define and prioritize the tasks required to manage enterprise security. Frameworks are also used to help prepare for compliance and other IT audits. The framework must, therefore, support specific requirements defined in the standard or regulation.

Organizations can customize frameworks to solve specific information security problems, such as industry-specific requirements or different regulatory compliance goals. Frameworks also come in varying degrees of complexity and scale. Today's frameworks often overlap, so it's important to select a framework that effectively supports operational, compliance and audit requirements.

To help manage the process, hereon is presented a list of the most known IT security standards.

CIS Critical Security Controls

The consensus Critical Security Controls [42] promulgated by the Center for Internet Security

enumerate prioritized technical safeguards and operational practices applicable across environments. Distinct from comprehensive risk analyses, the Controls focus exclusively on mitigating inherent weaknesses through inventory management, access restrictions, logging, defences, and testing. Partnering with preexisting governance improves risk remediation.

COBIT Frameworks

Originally oriented towards auditing IT risks in the mid-1990s, the Control Objectives for Information and Related Technologies (COBIT) framework [43] established by the Information Systems Audit and Control Association (ISACA) has since evolved to integrate business objectives and technologies. COBIT 2019, the current iteration, remains preeminent for assuring Sarbanes-Oxley compliance through aligned certifications and guidance.

ISO/IEC 27000 family of standards

The ISO/IEC 27000 [44] family of standards was developed by the International Organization for Standardization in partnership with the International Electrotechnical Commission to provide a flexible Information Security Management System framework applicable across public and private sector organizations of all sizes.

Chief among the standards are ISO/IEC 27001 and 27002, which define requirements and guidelines for establishing an Information Security Management System to facilitate auditing and compliance activities. ISO/IEC 27000 offers an overarching perspective while ISO/IEC 27002 specifies best practices for implementing controls.

Certification to the ISO/IEC 27000 family is granted via third-party assessments and ensures the proper operation of safeguards. The suite contains over 60 standards addressing diverse facets of information security such as cloud computing, business continuity planning, digital forensics, storage mechanisms, and healthcare informatics.

NIST Cybersecurity Framework

The National Institute of Standards and Technology developed the Framework for Improving Critical Infrastructure Cybersecurity (NIST Cybersecurity Framework or NIST CSF) [45] pursuant to Executive Order 13636 issued in February 2013. The NIST CSF aims to bolster the security posture of vital national systems including energy, water, food, communications, healthcare, and transportation—all sectors that necessitate resilient defences against threats such as state-sponsored actors due to their strategic significance.

Distinct from other NIST guidance, the NIST CSF prioritizes cyber risk assessment and management. Its security controls adhere to the five concurrent stages of the risk management lifecycle: identification, protection, detection, response, and recovery. Effective implementation of the NIST CSF demands executive sponsorship across both public and private organizations.

NIST Special Publication 800-53

The National Institute of Standards and Technology initially released SP 800-53 [46] in 1990 as the preeminent baseline of security controls for federal information systems. Ubiquitously leveraged across both public and private entities, SP 800-53 fueled the evolution of subsequent frameworks like the NIST Cybersecurity Framework. SP 800-53 provides guidance aligned with modern technologies including cloud computing.

NIST Special Publication 800-171

NIST SP 800-171 [47] gained prominence owing to mandates from the United States Department of Defense requiring demonstrations of alignment between contractor systems and its directives. Oriented towards safeguarding controlled unclassified information, SP 800-171 establishes fundamental security postures for third-party entities interacting with government programs. Organizations can demonstrate compliance with the more comprehensive SP 800-53 by first satisfying SP 800-171, benefiting smaller actors with constraints.

Controls included in the NIST SP 800-171 framework are directly related to NIST SP 800-53 but are less detailed and more generalized. It is feasible to construct mappings between the two standards if an organization must show compliance with NIST SP 800-53, leveraging NIST SP 800-171 as the starting point. This creates flexibility for smaller organizations as they can expand safeguards over time per additional controls in NIST SP 800-53.

2.4 Current challenges

The increasing size and complexity of software have led to an increase in the range of cyber attacks, as well as the risk of information exfiltration and data breaches. The common goal is to steal information and data by exploiting vulnerabilities within the code. This, in turn, results in users' privacy violations. This implies the need to identify and understand (at least) the most common threats in software security, disseminate security best practices, and address security from the earliest stages of software development [37] as a minimum effort to protect users' privacy.

Security should be a basic feature of software applications, guaranteed by the existence of — as an example — automated mechanisms for constructing complex and secure passwords rather than procedures for periodically renewing those passwords. Lack of system security can compromise privacy, so privacy becomes a proactive, integrative, and creative approach to strengthening security requirements, starting with the design of new systems and the protection of information assets (and data) in the case of existing ones.

It becomes necessary to consider and pursue privacy at all stages of the software lifecycle. Today we can identify several approaches that address security; however, they rarely consider the data privacy side of the problem. The same can be said for current privacy-oriented approaches. In the end, what emerges is that privacy and security are addressed separately from existing approaches.

The challenges that companies and developer communities face when trying to consider both privacy and security at the same time are many. In order to reach this goal, at least two macro-issues must be addressed:

1. Translate best practices for secure application development as well as data privacy into operational guidelines that can be traced back to code structures and software architectures;

2. Share security and privacy expertise within the development team. Privacy and security require specific skills that developers, even talented ones, often do not have. Therefore, there is a need to share and transfer knowledge effectively [48, 49].

2.5 State of the art

First of all, I investigated the literature relating to conventional software systems. The goal was to understand what has been done, what results have been obtained and what else could be done to contribute to the state of the art. Moreover, the work done in this field provides lessons learnt that I later kept in mind when dealing with AI-based software systems.

2.5.1 Privacy and Security in Conventional Systems

Privacy-oriented approaches

Considering what has been said so far, it is in the interest of everyone that privacy should be built as early as possible in the product development process rather than bolt-on privacy at a later stage in the design process. A privacy framework for businesses and policymakers proposed by the US Federal Trade Commission [50] states "*companies should promote consumer privacy throughout their organizations and at every stage of the development of their products and services. Companies should incorporate substantive privacy protections into their practices, such as data security, reasonable collection limits, sound retention practices, and data accuracy. Companies should maintain comprehensive data management procedures through the life cycle of their products and services*".

On the other side, Article 25 of the European Union GDPR [16] brings these ideas together and formulates the requirements of data protection by design and by default. Its definition of PbD places the responsibility on the data owner (or their controller) to implement technical and organisational measures to maintain user privacy before and during data processing [51].

To instruct developers to incorporate privacy into the design of software systems, there are several abstract well-established and widely recognized privacy practices. The most mentioned in the current literature are: Fair Information Practices (**FIP**) [52], Privacy by Design (**PbD**) [26] and Data Minimization (**DM**) [53].

DM essentially states that data should only be collected if it is related to the purpose of the application and should only be processed for the purpose for which it was collected.

FIP states that users should have access to and control over their data even after entering it into a system [52].

Cavoukian [54] laid the foundations of PbD practical implementation through seven principles, listed below:

- Proactive and preventive
- Privacy by default
- Privacy by design
- Full functionality
- End-to-end security
- Open visibility and transparency
- Respect for user privacy

Anyway, studies show that most developers still lack formal knowledge about privacy practices and that they are often led to try to incorporate privacy into software projects by proceeding tentatively. Sometimes, they find privacy contradictory to the functional requirements of the system and find it difficult to assess whether they have correctly integrated privacy into a system [55]. In an empirical study of 124 engineers, conducted to understand the motivations and impediments related to privacy and security engineering, Spiekermann et al [56] found that software engineers lack the time and autonomy needed to build successful ethical systems. Moreover, privacy and security standards in organizations are often inadequate or even opposed to privacy-oriented design, leading to conflict between engineers and their organizations.

As a consequence, in many cases, developers do not follow these practices when developing software systems. If software systems were designed to process data only for the purpose for which it was collected, and if users were given control over their data, we would not see events like the recent Facebook privacy scandal, where Cambridge Analytica accessed and processed the personal data of 50 million users without their consent. Facebook responded to the scandal by saying that strict measures would be taken to limit developers' access to user data and remove those who do not agree to be subjected to privacy controls. This makes it clear that developers play a central role in protecting privacy [57].

Privacy Engineering

Academics reacted to the difficulties presented in the previous section with Privacy Engineering.

Privacy Engineering (PE) is a nascent field in the realm of Software Engineering. Its purpose is the research and practice of systematic approaches to create privacy-oriented solutions across software systems and the SDLC [29]. Hansen et al. [58] define PE as an approach used to ensure that organizations provide sufficient protection to their stakeholders' personal data. One of the

most challenging aspects of PE is the fact that the concept of privacy is ”fuzzy” and difficult to define explicitly within the Software Development Life Cycle (SDLC) [59].

The approaches that researchers proposed and experimented with to achieve PE can be grouped into three main areas: Privacy Impact and Risk Assessment (PIRA), Model-based Design (MbD), and Privacy Requirement Engineering (PRE).

The following subsection presents some of the most cited approaches for each of these three main areas.

PIRA approaches

A Privacy Impact Assessment (PIA) is defined as a process that identifies and mitigates, with stakeholders' consultation, the potential impact on privacy caused by whatever initiative. PIAs are now mandated by, for example, the EU General Data Protection Regulation (GDPR). However, they do not precisely illustrate how a PIA should be performed. A PIA tends to focus more on legal and organizational aspects than technical details. In order for a PIA to be holistic and effective, it must be complemented by an appropriate privacy risk model that considers legal, organizational, social, and technical aspects: here comes the concept of risk assessment, which typically follows a step-by-step process of both risk identification and risk mitigation [60].

From the union of these two concepts comes Privacy Impact and Risk Assessments (PIRAs), that play an essential role in protecting the privacy of data subjects and managing privacy risks through systematic identification and mitigation [60]. One example is the CNIL approach, which includes a risk model that describes threats, vulnerabilities, and sources of privacy risks [61]. The CNIL methodology uses a semi-quantitative approach based on a predefined set of threats, together with their numerical index [60].

A similar approach is used in PRIAM [62], which is a privacy risk assessment methodology that collects information about seven components of a computer system, including stakeholders, data, sources of risk, privacy weaknesses, feared events, damage and the system itself. Each component has a set of categories and attributes that are used in risk calculations. These calculations are based on ”threat trees”, which represent the relationships between privacy weaknesses, feared events, and damages.

Alshammari and Simpson [60] extended the PRIAM methodology by introducing assessments that reveal weaknesses in privacy measures based on how easily they can be exploited and their severity. They also consider the value of personal data and how this affects the motivations of the source of the threat.

Model-based design approaches

Model-based Design, as the name implies, is based on models. A model is a systematic representation of an object or event in an idealized and abstract form; the act of abstracting eliminates certain details to focus on essential factors. A model provides a vocabulary for

discussing certain issues and is thus more like a tool for the scientist than for use in, for instance, practical systems development. [63]

LINNDUN is a systematic approach for privacy threat assessment designed for software engineers. LINNDUN is a mnemonic for the seven categories of high-level privacy threats: *Linkability; Identifiability; Non-repudiation; Detectability; Disclosure; Unawareness; and Non-compliance* [64]. LINDDUN is an approach based on Data Flow Diagrams (DFDs), which are used to model the flow of personal data through different processing events and areas of processors and controllers [65].

Al-Fedaghi [63] developed a framework that uses a flow-based model to conceptualize generic operations performed on personally identifiable information (PII): Flowthing Machine (FM), that defines six typical movements of PII within information systems: *Transfer, Processing, Creation, Release, Delivery and Acceptance.*

The key problem in PbD is the lack of guidelines on how to map legal data protection requirements into system requirements and components. Hoepman [66] analyzed both the PbD principles and the GDPR Articles and extracted the following Privacy Design Strategies to keep in mind for privacy design: *Minimise, Hide, Separate, Aggregate, Enforce, Control, Inform and Demonstrate.* These strategies are used both to design privacy-preserving systems and to assess the privacy impact of IT systems. In subsequent research [67], the design strategy named "Control" was extended with the creation of privacy patterns, which constitute a sufficient base of patterns with descriptions, relationships, and applicability.

Privacy Design Strategies represent an attempt to reduce the gap between "what to do" and "how to do it". To further improve them, in [68] the authors try to correlate and map the available strategies with the "Privacy Patterns" needed to implement them, but the results obtained are limited in scope and far from being used in practice. Privacy Patterns [69] represent a general solution to the most frequent privacy problems in software development. A further step in this direction was made by Suphakul et al. [70]: they proposed design patterns to include information about privacy principles addressed and relevant software models in the UML notations to use [71]. In [72], the authors propose a set of privacy process patterns for creating a clear alignment between privacy requirements and Privacy Enhancing Technologies (PET) [73], and encapsulate expert knowledge of PET implementation at the operational level.

One of the most comprehensive frameworks — based on PbD principle, strategies, privacy patterns and PETs — is the Privacy Oriented Software Development (POSD) [74]. This framework is based on a knowledge base that links all the previously mentioned concepts, across the entire SDLC, so that can be used by different kinds of stakeholders during the entire software development process. Moreover, in a subsequent work, this was provided with a graphical tool [2].

PRIPARE (Preparing Industry to Privacy by Design by supporting its Application in Research) [75] begins to highlight how privacy requirements can be incorporated into the SDLC. The study introduces a systematic methodology for privacy engineering, while Privacy-Friendly

Systems Design [76] incorporates privacy through the following steps: elicitation of privacy requirements, analysis of the impact on the process, and identification of supporting techniques.

Problem-based Privacy Analysis (ProPAn) [77] adopts a problem structure model to detect privacy threats. It uses privacy graphs, i.e. UML profiles that contain privacy requirements, and determines the stakeholders whose personal information requires protection [78].

He and Antòn presented an agent-oriented framework for modeling privacy requirements as the contexts and obligations of Role-Based Access Control (RBAC) entities and relationships, thus linking privacy requirements to organizational access control policies [79]. It includes a context-based data model for representing roles that have permissions to access data objects and privacy elements linked to these objects. RBAC is not supported by formal models but a guiding framework has been defined for expressing the relations between the roles and their restrictions on the organization's assets. Finally, a software tool has been developed, but it does not cover the whole method since it does not support role analysis.

Mai et al. [35] proposed a modeling method based on extending the use case diagrams belonging to UML. Their main motivation was to enrich these diagrams with security and privacy adding a limited number of extensions. An important aspect of this work was the modeling of threat scenarios and their possible mitigation processed in a "structured form". The main limitation of this work is that extends only the use case diagram, which is the most general UML diagram. For this reason, it is difficult to consider this work for complex environments (such as the IoT). In fact, Aufner [80] stated that even if threat models and their implementation to increase security have been deeply researched in social networks, there is a lack of research in the IoT field.

Privacy requirements engineering approaches

Privacy Requirement Engineering (PRE) is essentially based on the elicitation of non-functional requirements: privacy requirements, which are considered a key point to developing a privacy-friendly system. Engineering privacy requirements in a disciplined approach is an essential step in building privacy-preserving software systems.

Examples of techniques relating to this approach are the adaptations of the Security Quality Requirement Engineering (SQUARE) process [81].

SQUARE is a comprehensive process developed for eliciting, categorizing, and prioritizing security requirements within an organization [34]. Its aim is to integrate security requirements at the early stages of software development. SQUARE advocates nine steps which are: *(1) agreeing of all the stakeholders on common definitions; (2) identification of security goals; (3) developing artifacts; (4) perform risk analysis; (5) selection of an appropriate elicitation technique; (6) elicitation of security requirements; (7) categorizing requirements; (8) prioritizing requirements; and (9) inspection of the final requirements.* Each step has some exit criteria that have to be fulfilled before going to the next step, although some of the steps can be performed in parallel.

Several researchers have adapted the SQUARE process to incorporate privacy requirements engineering.

One of the results is eSQUARE [82], a web-based tool developed based on Z language for modeling and checking the consistency of security requirements; it also contains a metric developed for measuring the security requirements and was aligned with the SQUARE methodology. In eSQUARE, the authors have integrated privacy requirements in the SQUARE methodology in order to identify privacy risks apart from security risks.

Another SQUARE adaptation is that of Bijwe and Mead [83]. They introduced in SQUARE the use of the Privacy Requirement Elicitation Technique (PRET), a computer-aided approach that uses a questionnaire to extract information from engineers and stakeholders. The proposed tool contains a database of privacy requirements that generates results based on the input from the questionnaire.

However, the work in [84] finds that integrating PRET into SQUARE is not an alternative to other privacy risk assessment methods and therefore proposed the addition of Health Insurance Portability and Accountability Act (HIPAA) and PIA for privacy risk and impact assessment to SQUARE.

Lai and Zhang et al [85] developed 2-SQUARE, a software that provides flexible guidelines for workflows and business processes. It aims to facilitate the communication of privacy requirements between engineers and stakeholders.

Similarly to what has been done for SQUARE, in literature there are several adaptations of the STORE methodology. The Security Threat Oriented Requirements Engineering (STORE) is a methodology focused on security threats that can be used by a software development enterprise to develop effective and efficient security specifications [86]. This approach identifies and gives priority to all stakeholders based on their importance. Since it is a detailed sequential procedure for managing and documenting security requirements, the security expert or requirements engineer cannot completely forgo as well as swoop among the 10 phases that compose the entire process. However, the STORE methodology is capable of eliciting only security requirements.

Hence, Ansari et al. have extended the STORE process as P-STORE methodology for incorporating the effective and efficient privacy requirements through a sequential procedure. P-STORE's ten steps are: *Identify privacy goals, Identify and prioritize stakeholders, Agreed-upon privacy goals, Asset identification, Privacy attack analysis, Privacy threat identification and categorization, Risk evaluation and prioritization, Privacy requirements elicitation, Privacy requirements validation, Privacy requirements specification document.*

Leaving aside adaptations, hereon are presented other PRE approaches.

The PRiS method [87] considers privacy requirements as organizational goals and describes the effect of integrating privacy requirements within business procedures in order to identify the most suitable system architecture. PriS is a security requirements engineering method, which incorporates privacy requirements early in the system development process. It considers privacy requirements as organizational goals that need to be satisfied and adopts the use of

privacy process patterns as a way to *(a) describe the effect of privacy requirements on business processes; and (b) facilitate the identification of the system architecture that best supports the privacy-related business processes.* Pris models privacy requirements as a special type of goal, the *privacy goal*, which constraints (have an impact on) the causal transformation of organizational goals into processes.

In these methodologies, and also in PRIPARE, privacy is addressed during construction or early software design activities instead of in all phases.

The STRAP method (STRuctured Analysis of Privacy) [88] was developed with the objective of eliciting and analyzing privacy requirements during the system design phase. In STRAP, privacy requirements are represented as vulnerabilities. STRAP builds a goal model that represents all functional requirements and vulnerabilities have the form of obstacles between the goals and the subgoals in the goal model. The method's way of working comprises of the following steps: *a) Analysis, b) Refinement, c) Evaluation and d) Iteration.* STRAP drawbacks are that is not supported by formal models and no software tool has been implemented to guide the designers and for the graphical representation of the method's models.

Meis [89] presented the Problem-based Privacy Analysis (ProPAn) for privacy requirements engineering, i.e. an aspect-oriented requirements engineering methodology. ProPAn enables customized modeling and incorporation of the intersection of oral features and functionality into the requirements specification of a software system. ProPAn attempts to address existing problems including the specification of functional requirements of a module through constructive feedback. The author has also provided a categorization of privacy requirements extracted and mapped to several other privacy concepts.

Security-oriented approaches

There are numerous frameworks in the literature focused on security. Hereon are presented the most cited and used by the industry, other than the already cited SQUARE and STORE.

Adam Shostack developed the STRIDE framework for threat modeling, which serves as a support for CIA — confidentiality, integrity, and availability: while the CIA triad defines the three pillars for designing secure systems, STRIDE presents six types of threats relevant to CIA.

STRIDE specifies six areas in which attacks on the security of a system can be classified; the areas are *Spoofing, Tampering, Repudiation, Information disclosure, Denial of service and Elevation of privilege* [90]. These threats are nothing more than the opposite of the properties that characterize a secure system: *authenticity, integrity, non-repudiation, confidentiality, availability, and authorization.*

Spoofing: A set of threats that violate authentication protocols, allowing a malicious user to use an identity other than the one he actually possesses. Spoofing can occur at any level of the ISO/OSI stack.

Tampering: data circulating within the system are altered by a malicious external user.

Repudiation: An attacker may refuse to take responsibility for a certain behavior, but this cannot be proven due to the absence of evidence, such as system logs.

Information disclosure: information is exposed to parties that do not/should not have access. Information loss and data breaches are common examples of information disclosure.

Denial of service: The system/information is not available to a legitimate user who is requesting it.

Elevation of privilege: The malicious or legitimate user gets more privileges over the content than he is entitled to.

The i* modeling framework was developed to support requirements analysis and design using the agent-oriented approach. It tries to model the dependency relationships among various actors using the concept of actors, agents, and goals [91]. Various extensions of the i* model have been proposed to model security and privacy concepts [82].

Secure i* [92], an extension of the i* model, analyzes security trade-offs among various stakeholders and the system to be built. It tries to align the security requirements with other requirements of the system. It is based on a general model that considers important security concepts like goals, tasks, assets, threats, and resources. A meta-model that aims to capture the act of attackers and the counteract of the victims using i* model is discussed in [82]. It is applicable in all the phases of SDLC and covers the three CIA goals.

Tropos [93] approach is based on the agent-oriented software engineering (AOSE) paradigm and i* modeling framework. The process involved in this approach is modeling the actors, modeling dependency between the actors, modeling trust, and refinement of goals.

A lot of Tropos extensions were proposed over time. For example, Secure Tropos-SPL [94] has been developed for modeling security requirements in the context of software product lines (SPL).

An extension of the Secure Tropos to formally model and optimize security mechanisms in alignment with security requirements has been discussed in [95].

A risk-aware Secure Tropos, an extension of the Secure Tropos model in alignment with information security risk management concepts, is discussed in [96].

A Tropos Goal-Risk framework for modeling and assessing security risk of critical systems based on trust relationship among actors is proposed in [97]. This framework uses trust as evidence to assess risk, evaluate the alternative goals, and analyze possible countermeasures for mitigating the risk based on the organizational setting of the system.

These were only a small part of the existent Tropos extensions. Now, we move further to analyze frameworks not related to Tropos.

Misuse Case is a modeling tool that models the behavior of actors hostile to the system under construction [98]. It is used along with the standard UML use case diagram in order to model the behaviors not desirable in the system [82]. A misuse case diagram represents threats and attackers along with a directed association between them. Various relationships used in the diagram: threatens, includes, and extends. A Misuse case diagram only captures the

basic overview of threats, and hence, a template has been proposed to give a detailed textual description of threats and attackers.

The idea of using patterns has been proved to be very valuable in the area of software engineering. They give an insight into a class of problems and help in designing software by providing an accumulated body of knowledge, so one does not have to start right from scratch. Strictly correlated to patterns there are the Problem Frames, introduced by Jackson [99], which define a problem intuitively using an already defined class of problems in terms of its domain, characteristics, interfaces, and requirements.

Hatebur et al. [100, 101] introduced two new problem frames, known as Security Problem Frames (SPF) and Concretized Security Problem Frames (CSPF). SPF help to fulfill the security requirements of the system. They refer to the security problems being solved without recommending a solution. The solution space is defined by concretizing the problem frames with generic security mechanism (like encryption techniques for secure data transmission), by converting SPF into CSPF. CSPF consist of preconditions and postconditions in formal notations that have to be fulfilled in order to create an instance of SPF. The authors have described a three-step methodology formodeling security-related problems. The first step consists of decomposing the problem into subproblems until, and a specific security problem frame is found accordingly. In the second step, generic security mechanisms are introduced to solve the security problems by concretizing a SPF with a known security measure. Finally, the last step introduces specific security measures and generic sequence diagram to achieve the perfect solution to the security problem.

2.5.2 Privacy and Security in AI-enabled Systems

Most of the problems already addressed in conventional systems should be kept in mind while developing AI-enabled systems. In addition, new attack categories and techniques emerge for AI-enabled systems, which should carefully be analyzed since few mitigations exist nowadays.

The biggest concern related to AI-enabled systems is that, in most cases, a failure in the underlying algorithm may have an impact on human lives, not only in terms of death but also fundamental rights: dignity, freedoms, equality and solidarity, citizens' rights and justice [102].

The novelty in AI-enabled systems is that a small perturbation in the data (inputs or other data consumed by the algorithms) might cause a drastic change in the system's decision.

This is why, even if AI-enabled systems are one of the fastest evolving fields out there and every day a number of new and useful tools are created whose aim is creating functional and effective systems, an emerging trend is some new tools are being focused on offering functionalities that go beyond simply building new and powerful systems; instead, they offer new perspectives by focusing on different aspects.

All these aspects are presented in the next sections.

Model Privacy

The first presented aspect is **model privacy**: this topic is receiving a lot of attention also outside the technical communities in last years, especially after the introduction of legal policy frameworks like the GDPR [39] and the next-coming AI Act [103]. Anyway, researchers and communities addressing the topic of privacy in these systems seem to be a niche: this is dangerous because there are mainstream attacks which can cause a data breach also in AI-enabled systems, resulting in loss of interest in the systems by the potential users.

It has been observed that a lot of individuals are concerned about the governance of AI-enabled systems and the prevention of algorithmic and data harms [104]. Also, GDPR is seen as a turning point for achieving better governance in this field. Anyway, the GDPR provides guidance on the use of algorithms only in limited cases; most of its guidelines relate to the governance of personal data but algorithms have traditionally been seen as intellectual property as well [104].

These problems arise from the fact that the process of feeding an unconventional system with training data is not one-way, and hereon there are the motivations.

When one wants to train a system, usually starts with some (potentially sensible) training data, composed by some data features X and corresponding class labels y. Chosen an algorithm, e.g. a neural network (NN), the training data is used to make the model learn a mapping/pattern which binds X to y. This mapping should be general, so that the model is also able to predict the correct labels for unseen data X'.

Now, the problem emerges from the obvious fact that, in order to learn a mapping from specific features to the corresponding labels, the system needs to ”remember” (learn, store) in its parameters some information about the data it was trained on. Otherwise, it would not be able to output correct conclusions about new and unseen data.

Based on the sensitivity of the data used for the training, the fact that some information about data is stored in the system parameters may cause privacy problems: someone able to access the system may deduct information about the training data. This demonstrates why some AI models should be legally classified as personal data.

Starting from the subset of attack techniques described in the next subsections of this essay, Veale et al. [104] explored different rights and obligations this action would trigger and their utility, as well as they posit future directions for algorithmic governance and regulation.

One may argue there exist many other privacy attacks for AI-enabled systems not addressed in this thesis, for example the ones mentioned in [105]: they are excluded because a lot of these attacks are too complex to be realized in a real context and/or need in-depth knowledge of the policies running in the organization.

In the following paragraphs there are various attack techiques that emerge from the literature regarding privacy in AI-enabled systems.

Membership inference attack

Using this technique, an attacker creates random samples and sends them to a victim AI model, usually served on a cloud service — this deployment architecture is usually known as Machine Learning as a Service (MLaaS). Based on the confidence score the model returns, the attacker tunes the samples' features and re-sends them to the model. This process continues until the victim model outputs a very high confidence score: in this case, the sample is identical or very similar to one of the samples used to train the victim model.

After gathering enough high-confidence samples, the attacker can use the handcrafted dataset to train a set of "shadow models" able to predict whether a data record is part of the victim model's training data or not. This creates an ensemble of models whose outputs can be further used to train a "meta-model", whose goal is to predict the membership inference of a sample: this final model can predict whether a sample was included in the training dataset of the victim model.

The researchers found that this attack was successful on many different AI services and architectures publicly accessible.

Such attacks are very common due to the growing interest in using MLaaS; this facilitates the attackers in developing and launching membership inference attacks over the same services.

The first and most famous attack in this category was proposed by Shokri et al. [106] at the 2017 IEEE Symposium on Security and Privacy; this attack worked on all major cloud-based machine learning services, Amazon and Google services among others. What the authors did was simply concretely reproduce the steps of the membership attack: in their work, they describe a malicious actor (an attacker) who creates random samples to be sent to a target machine learning model served on a cloud service. The attacker needs to be able to send each sample to the model and receive a correlated output. Based on the confidence score the model returns in the output, the attacker tunes the sample's features and re-sends it to the model. The process continues until the victim model outputs a very high confidence score. At this point, the sample can be considered identical or very similar to one of the samples used to train the model. After some time, the attacker will obtain a large number of high-confidence samples, which construct his dataset. Then, the attacker uses this handcrafted dataset to train a set of so-called **shadow models** able to predict whether an input sample was part of the victim model's training data. These shadow models represent an ensemble of models: their predictions compose a vector used to train a **membership inference attack model**: this final model can predict whether a data record was included in the training dataset of the victim machine learning model.

Later, de Arcaute et al. [107] studied the impact of membership inference attacks on traditional machine learning algorithms and showed that many ensemble models, most of all Extra-Trees, are vulnerable to this attack. This information can be used by designers (and all stakeholders involved in the creation of a model) as an additional guideline useful in selecting the machine learning algorithm so that privacy-protection is incorporated as a design goal from

the initial phases of the development lifecycle.

The existent mitigations to these attacks are hereon presented.

Regularization-Based Defense Methods

Shokri et al. [106], after proposing for the first time the membership attack, investigated four defence methods. The first is based on the *restriction of the output vector* (containing the predictions) to the top k classes - where a smaller k results in less information provided. The second one is based on providing confidence score vectors with *less accurate precisions*, e.g. rounding or truncating the probabilities within the vector down to d floating point digits; of course, a smaller d entails less information provided. The third one is based on *increasing the entropy of the confidence score vector* by using a softmax function and a temperature t to compute the output of the logits vector. The fourth and last one is based on using L2-norm standard r*egularization over the scores* in the output vector.

In [108], Nasr et al. made a proposal based on a min-max privacy game, whose solution is achieved by training a model using an adversarial process that minimizes both the prediction loss and the maximum gain of the attack. The obtained model is able to provide both membership defence and strong regularization capability.

Salem et al. [109] also proposed two defence methods against membership inference attacks. The first one is called "dropout" (in fact works like the dropout layer in neural networks) and is based on a random deletion of a prefixed proportion of edges from a fully connected neural network model in each training iteration with the goal of avoiding overfitting. The second method is called "model stacking" and is based on ensemble learning to construct the final model: two models take the original training data while the third is trained with the conference score vectors produced by the previous two models. This should contribute to avoiding overfitting.

Adversarial Example-Based Defense Methods

Jia et al. [110] proposed a defence method named "MemGuard" based on the idea of adding noise to each confidence score vector in order to make it appear like an adversarial example[24]. In their work they describe the entire process of adding noise as an optimization problem and also the developed algorithm used to solve the problem based on gradient descent.

Deep Neural Network-Based Defense Methods

Yang et al. [111] proposed a purifier model whose goal is to transform the output vector of a model: it takes a confidence score vector as input and reshapes it. Substantially, is based on the encoder-decoder model: the encoder maps the confidence score vector to a latent representation while the decoder subsequently tries to reconstruct the initial confidence score vector.

Differential Privacy-Based Defense Methods

[24]More details about adversarial samples are provided in the next paragraphs.

Differential privacy has been used a lot to preserve the privacy into deep learning models [112] [113–115]. The core concept behind this defence technique is adding noise to the data. This noise can be added in five different places in an algorithm: [116] proposes the adding to the input datasets, [117] in loss functions, [113] in gradients, [118] in the weights of neural networks, and [119] in output classes.

Recently, also local differential privacy has been used as mitigation. For example, Kim et al. [120] adopted the Gaussian mechanism to realize local differential privacy in federated learning models. They also analyzed the trade-offs between user privacy, global utility and transmission rate, since a larger noise leads to stronger privacy but lower utility and higher transmission rate.

Anyway, it must be noted that most of the presented techniques constitute a defence only against membership inference attacks, and the differential privacy methods have been developed for general privacy preservation rather than to protect against a specific inference attack. Some methods, moreover, require a re-train of a neural network, which is not an optimal strategy if time efficiency is a strict requirement. Following these motivations, in [121] was proposed a strategy based on modifying and then normalizing the output confidence score vectors with a differential privacy mechanism: no new models need to be trained, so this is a very time-efficient method. In addition, the order of scores in the confidence vectors is preserved thus no tradeoffs have to be made.

Limits of membership inference

The severity and risks associated with membership inference attacks largely depend on the applications and the type of data used for training the victim model: in applications involving complex image and speech classification tasks, the computational efforts required to generate training data implicitly reduce the severity of the attack. On the other hand, in some human-centric applications based on tabular data — such as education, finance, and healthcare — which can be easily generated, membership inference attacks may pose severe implications.

Membership inference is also highly associated with ”overfitting”, usually a consequence of bad design and training. An overfitted model performs well on its training examples but poorly on unseen data. Two reasons which lead to overfitting are (i) having too few training examples and (ii) running the training process for too many epochs. The more overfitted an AI model is, the easier it will be for an attacker to successfully conduct a membership inference attacks against it. As a consequence, a model able to correctly generalize on unseen examples is implicitly more secure against membership inference.

Model Inversion attacks

This attack was first proposed by Fredrikson et al. [122]; the core idea is to use a (new) model to extract representations of the initial training data used for the victim model. In other

words, the goal of this new model is to iteratively query the victim model with the goal of finding (or reconstructing) the inputs used to train the victim model.

The low-level philosophy behind a model inversion attack is very simple: reverse-engineering a model by using the gradient in a trained network to tune the weights and obtain the features for all classes. It is possible to create and start with a prototype sample for the classes for which not prior information is available. Moreover, this type of attack demonstrates that any accurate neural network may leak information on the different classes on which has been trained. A number of works has shown that artificial data generated by a Generative Adversarial Network (GAN) are very similar to the victim's training data. As a consequence, the information obtained by a model inversion attack may "reveal more private information about the training data when compared to the average samples" [123].

Generally, this type of attack works in a white-box model (i.e. the attacker has access to the trained model, in particular to the model parameters); anyway, it also can used in a blackbox environment (i.e. the attacker is only allowed to query the prediction model on his/her inputs through an appropriate interface) with lower effectiveness.

The first and most famous attack in this category was proposed by Fredrikson et al. [122]: they used a trained classifier to extract representations of the initial training data used in the victim model.

In particular, a face classifier trained on black and white images belonging to 40 different individuals was used. The training data features X are the individual image pixels, which can assume numerical values in the range [0,1]. Since the number of different pixel value combinations over an image is very large, it is infeasible to realize a brute-force attack to obtain a reconstruction over all possible images with the goal of identifying the one(s) that most likely belong to the training dataset. Therefore, the authors proposed a different approach: given m different faces and n pixel values per image, the classifier can be expressed as a function:

$$\int : [0,1]^n \rightarrow [0,1]^m$$

The output of the classifier is a vector that represents the probabilities of the image to belong to each of the m classes.

The authors also defined a cost function *c(x)* related to the above function to realize the attack: selected a candidate solution image *x0*, the gradients of the cost function are calculated. Then, the gradient descent technique is applied and *x0* is transformed iteratively in each epoch i according to the gradients in order to minimize the cost function. When a minimum is found, the transformed *xi* can be returned as the solution of the model inversion.

At the end of the technique algorithm, the minimal costs and the corresponding *xi* are returned. If, as happened to Fredrikson, each individual corresponds to a separate class label, the model inversion can be used in order to reconstruct concrete faces.

Later, Want et al. [124] proposed a technique based on an ensemble able to estimate the distribution of original training data by exploiting a newly trained generator constrained by an

ensemble of models trained on shared data. They gained visible improvements in the quality of the generated samples compared to using only a single model rather than an ensemble.

Khosravy et al. [125] demonstrated the use of Deep MIA, an integration of deep generative models in MIA, and α-GAN initialized with a face-based seed (α-GAN-MIA-FS). As experimentation, they used a pre-trained deep generative model with the capability of generating a face image starting from a random feature vector. The goal was to narrow down the image search space to the feature vectors space, which has much lower dimensions. As a consequence, the model inversion attack can be more efficiently conducted because the searches can be done on a low-dimensional feature vector.

Since initially most existing defence methods only protect against membership inference and not also from model inversion attacks, and methods that can combat both types of attacks require a new model to be trained — which may not be time-efficient —, Ye et al. [121] proposed a new defence method based on differential privacy able to handle both attacks in a time-efficient manner by tuning only one parameter: the privacy budget. The core idea is modifying and normalizing the confidence score vectors with a differential privacy mechanism which preserves privacy but obscures membership and reconstructed data. Then, this method can also guarantee the order of scores in the vector and avoid any loss in classification accuracy. Finally, the experimental results demonstrated this method is an effective and timely defence.

Moving to a different kind of defence measures, we can generally say encryption can block the adversary's knowledge and maintain black-box condition, thus encryption can be generally considered effective against white-box attacks like model inversion.

One may think another effective defence measure is differential privacy. This is wrong because in model inversion an attacker just uses the output of a model to infer certain features of the training data. According to the differential privacy concept, this does not necessarily lead to privacy breaches. Consider a face recognition system, where a single person is represented by an output class of the model. Since all training images for this class are various photos of the same person, an attacker can launch a model inversion attack by handcrafting an artificial image which is composed of the average information derived from the available person's photos. In most of the cases, this average will be identified as that person. So, this average of features can represent the entire output class but does not represent a particular member of the training data set.

Therefore, the model inversion attack is even effective with differential privacy in place because addresses the parameters of the model and the granularity is set at the record/instance level. Once the model becomes accurate, it will contain the noise added to the learning parameters. However, as long as the model can accurately classify the class and generate representations of that class, the attack can be considered successful.

Model Security

The second aspect dealt with in this essay is model **security**: this topic is one of the main concerns even in AI-enabled systems. The various attack techniques that exist are quite different — such as adversarial attacks, model extraction, backdoors, trojans, membership inference, and model inversion — and will be investigated in the following paragraphs.

Adversarial attacks

The core idea behind this technique is crafting fake data (i.e. "DeepFake(s)") belonging to different domains — such as text [126], images [127], audio [21], network signals [128] —, known as adversarial examples, that in some way make the model vulnerable when included in the training set.

Using a more technical formalism, we can use X for referring to benign input data which is classified as class 1 by the model M: the goal of this attack is to find a function F able to generate a malign input data X' (say F *poisoning function* where $F(X) = X'$) so that X' is classified as belonging to another class (e.g. 2) by the same model M but the difference between X and X' cannot be identified by a human-eye. This attack is called **un-targeted adversarial attack**.

The **targeted adversarial attack** requires a target class Y: the goal of the function F is to find any benign input so that model M becomes biased towards Y in prediction; in other words, the model M predicts exactly class Y instead of other classes.

Adversarial attacks may be classified also based on the amount of knowledge the attackers have about the target model (i.e. victim model): there exist white-box, grey-box, or black-box attacks. In **white-box attacks**, attackers have full knowledge about the targeted model architecture (they know the source code and the weights). This eases the crafting of poisoned data used to fool the system. In **grey-box attacks**, the attackers may have some information about the overall structure of the model. In **black-box attacks**, all they have is just access to use the model [129].

In [130] a more detailed adversarial attacks taxonomy is presented. Fig. 2.3 shows a common adversarial attack on an image classification model: the perturbation added to the image makes the model change the prediction.

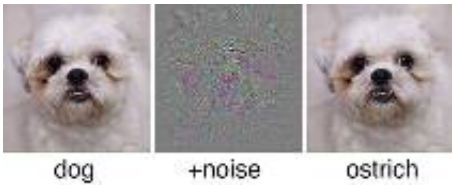


Figure 2.3: A graphical example of adversarial attack with the corresponding predictions [9].

There is not just one answer to the question "why adversarial attacks exist". The first

reason was provided by Szegedy: in his paper [9] he argued that the cause may be too much non-linearity and poor regularization of neural networks. Later, Goodfellow [131] argued that adversarial examples occurred due to too much linearity, the opposite of the previous theory. His main point was activation functions like ReLU and Sigmoid are basically straight lines in the middle (where also it is needed to prevent gradient explosion or vanishing). So inside a neural network there are tons of linear functions, all perpetuating one another's input, all in the same direction. If tiny perturbations are added to an input (a few pixels in different positions in the original input), the perturbations accumulate into massive differences on the other end of the network and the output is a paradox.

Another hypothesis, maybe the most shared today, is the tilted boundary [132]: the authors argue that since the model never fits the data perfectly (otherwise test set accuracy would always be 100%), there will always be a margin for the adversarial in the input, that exists between the boundary of the classifier and the actual sampled data. They also empirically debunk the previous two approaches.

There are also two mathematical explanations:

- Lack of sufficient training data [133],
- Computational intractability of ever building robust classifiers [134].

Finally, there is a recent paper from MIT with another proposal: Ilyas et al. [135] argued that adversarial examples are not caused by a bug but are a feature generated by how neural networks see the world. In other words, the fact that humans are limited to 3 dimensions and can't distinguish noise patterns is not a valid motivation to exclude noise patterns from the set of valid features.

In the same paper, another really interesting phenomenon related to adversarial examples is addressed: their transferability. This means adversarial examples can be re-utilized among both different models and different datasets. For example, after training a model to distinguish between dogs vs cats, one could be able to fool a completely different model trained on hotdogs vs not hotdogs with the same adversarial noise. This is logical: since the noise is actually a feature, it makes sense that other images in the image realm (be they cats, dogs, hotdogs or not hotdogs) all share it.

As the reader may be thinking, adversarial is not a "new" phenomenon. Battista Biggio, one of the most active and famous names in Adversarial ML, in his paper [136] stated that attacking machine learning algorithms has already been done in 2004. Then, adversarial attacks were used against spam filtering algorithms, in order to show that linear classifiers could be easily fooled with just a few handcrafted changes in a spam email.

Anyway, only later these attacks have been applied in the context of deep learning. For example, in 2013 Christian Szegedy from Google AI — while working on understanding how neural networks learn and think — discovered a different common property: all neural networks can be easily fooled by small perturbations [9].

Before presenting the possible attack techniques, it must be pointed out that the attacker's knowledge about the target system (named "capability") is important. The more he knows about the target model and how it is built, the easier is for him to attack the victim.

Overall, there are five types of adversarial attacks:

- Based on gradients,
- Based on the use of confidence scores,
- Based on the use of hard labels,
- Based on the use of surrogate models,
- Brute-force attacks.

Gradient-based attacks

These attacks by definition require access to the model's gradients and therefore are considered white-box attacks. Of course, these are the most powerful attacks, since the attacker has access to the internal details regarding how the model thinks (its gradients); so the attacker is able to mathematically optimize the attack.

This is also the technique that works best for attacking hardened models. Lots of works have shown that, if the attacker has access to the model's gradients, he will always be able to craft a new set of adversarial examples to fool the model. The most important aspect is that, security through obscurity aside, it's actually really hard to defend against these attacks.

The three most powerful gradient-based attacks according to the state of the art are:

- EAD (based on the use of L1 norm as distortion metric) [137],
- C&W (based on L2 norm) [138],
- Madry (based on Li norm) [139].

Confidence score attacks

Confidence score attacks exploit the classification confidence, provided by the model as output, to firstly estimate the gradients of the model and then perform smart optimization to the basic gradient-based attack. Since this approach does not require any knowledge about the interal structure of the model, it is considered a black-box attack.

Three are the most powerful confidence-based attacks according to the current state of the art:

- ZOO [140],
- SPSA [141],
- NES [142].

Hard label attacks

Hard label attacks just need the label provided by the model as output (“cat”, “dog”, “hotdog”) and not also the confidence scores. This makes the attack less optimized but more realistic (most of the public endpoints do not provide the classification score). The most powerful attack in this category is Boundary Attack [143].

Surrogate model attacks

Surrogate model attacks are based on the same logic of gradient-based attacks; the difference is that there is an extra initial step. Since the adversary has not access to the model internal details, with this approach he tries to first rebuild the victim model. This can be done in two different ways:

- If the target model is distributed like an oracle, the attacker can infer this information by repeatedly querying the endpoint and taking note of the input-output correspondence.
- If the target model is not open to the public and locally used for a standard classification task (e.g. object detection), the attacker can just guess the architecture and data the model is trained on (maybe with the help of OSINT techniques) and use that to build a copy.
- If the attacker has no information at all, he can exploits the transferability property and simply take any off-the-shelf image classifier to produce imperfect - but still functional - adversarial examples.

Brute-force attacks

Finally, there are Brute-force attacks. Like the classical definition of "brute force", here no optimization at all is used to generate adversarial examples. Practical techniques used are simple, like:

- Randomly rotating or translating the image [144],
- Applying common perturbations [145],
- Adding Gaussian noise with large SD [146].

After presenting the attack categories, now the essay will move on to the defences.

Defending against adversarial examples is hard. Two broad categories of defences exist: empirical defences and formal methods.

Formal methods

Generally speaking, Formal methods are a mathematical technique used to guarantee the robustness of conventional software/hardware systems. In most software which composes an AI-enabled system, one mistake in the design or development might lead to downtimes and/or

bad customer reviews, but (almost) never to human death. In some industry fields, however, such flaws are unacceptable. Producing millions of chips and then realising some parts of the chip are flawed is unacceptable. Or, again, a plane cannot be launched in the air unless a mathematical verification is done to demonstrate every component will work as intended.

What a formal method does is attempt every possible scenario and see the results. In the context of adversarial attacks this results in trying to generate every possible adversarial example within a certain range of perturbation. For example, consider an image with just two gray-scale pixels — say 180 and 80[25]; Then suppose a perturbation in the range of 3 in each direction. This results in 121 combinations to try out: each one should be submitted to the model to see and analyze the result.

This could be an acceptable number but, of course, the is exponential growth which rapidly makes the number very large.

Clearly, the problem with formal methods is they are not computationally cheap (or, even, impracticable). In fact, state-of-the-art formal method techniques can verify a network with no more than two or three layers. So, at the moment, formal methods are far from being used in a practical context. Anyway, there are still some good papers in the literature about formal methods, like [147], [148] and [149].

Empirical defences

On the other side, there are empirical defences. As the name suggests, these rely on technical experiments to test and demonstrate the effectiveness of a defence. As a basic example, one could attack a model twice: first a normal, without any defence, version and then a hardened one; finally, observe and compare the ease in fooling both of them (the hardened model should perform better). The most immediate consequence is that while formal methods try to evaluate every possible scenario - to verify no adversarial examples can fool the system -, empirical methods evaluate just a small subset of possible attack cases.

Also for the empirical method, there are popular attacks:

- Adversarial training,
- Gradient masking,
- Input modification,
- Detection,
- Extra class.

Adversarial training

Adversarial training is the most studied and effective defence according to the current state of the art. As the name implies, in adversarial training the defender re-trains the model including

[25] `https://towardsdatascience.com/evasion-attacks-on-machine-learning-or-adversarial-examples-12f2283e06a1`

also adversarial examples in the training dataset, correctly labelled. This way the model learns how to ignore the noise and only select the most "robust" features.

Clearly, the main drawback is the model becomes robust only against the crafted adversarial samples. So, if an attacker then mounts an optimized attack using a different approach, or an adaptive one (i.e. a white-box attack on a hardened model), he is still able to fool the classifier as if no defences were in place.

Also, continuously retraining the model with newly forged adversarial examples could lead to a situation in which so much fake data was inserted into the training set that the boundary the model learns becomes useless since is no longer able to correctly classify either genuine or malicious samples.

Anyway, using adversarial training still makes it harder for the attacker to bypass the classifier. Some practical techniques to implement this approach are:

- Ensemble AT [150],
- Cascade AT [151],
- Robust optimization approach to AT [139],
- AT via spectral normalization [152].

Gradient masking

Gradient masking could be considered a non-defense because belongs to the "security-by-obscurity" class of defenses. Even if for a long time the literature has considered effective this approach - since gradients are needed to compute powerful attacks on models -, hiding the gradients actually does not protect a model. In fact, *Defensive distillation* [153] has been very popular but its falseness was quickly proved [154].
The main reason behind the gradient masking ineffectiveness is, again, adversarial examples' transferability property. Even after hiding the gradients of a model, one attacker can still build a surrogate, attack it and then re-utilize these examples.

Input modification

Input modification basically consists of "cleaning" the input before passing it to the model, in order to remove the adversarial noise. Examples of technical implementations are:

- denoising solutions (autoencoders [155] and high level representational denoisers [156]),
- color bit depth reduction [157],
- smoothing [157],
- reforming GANs [158],
- JPEG compression [159],

- foveation [160],
- pixel deflection [161],
- general basis function transformations [162].

Detection methods

Some detection methods are based on input modification: once an input has been cleaned, its prediction can be compared with the original prediction; if there is a big difference between them, it is likely that the input has been tampered with by an attacker. Dongyu et al. [163] present some practical examples.

There are also different detection methods. For example, Metzen et al. [164] trained a separate detection network whose only job was to decide if the input was adversarial.

Detection defences are employed with the goal of computing and examining raw statistics at various points in the input query process:

- On input itself [165],
- On convolutional filters [166],
- On ReLU activations [167],
- On logits [168].

The strength of input modification & detection methods lies in the fact that can be applied to an already trained model and do not need the data scientist to go back to the origins of the model.

Extra (NULL) class

The last approach is the extra (NULL) class, proposed by Hosseini et al. [169]. The idea is simple: instead of forcing the classifier to provide a label even when it has no knowledge about it, give it the possibility of abstaining.

Remarks

The conclusion behind this overview is that empirical defences are imperfect but work, sometimes even with only a few lines of code. Even if new defences are often quickly demonstrated to be ineffective, one important aspect is at least trying to make just hard enough the attacker's job.

Data poisoning attacks

As the name states, here attackers intentionally send manipulated data to the system, e.g. an image with incorrect labels, so the underlying model gets re-trained with this fake data and its performance gets degraded. In this technique of attack, of course, attackers need to have control over or access the victim's training data [170].

We can take smart cities as an example of context [19]: here crowd-sensing is the primary data source for smart services involved in a number of areas like public and private transportation, energy management and pollution monitoring [171]. However, the other side of the coin is that crowd-sourcing is highly susceptible to data poisoning attacks [172, 173]; moreover, in some situations, manipulated data appear to be very reliable so that they are hard to identify [174,175].

In more critical fields, this technique of attack may have highly severe impacts: in an experiment on around 17,000 healthcare records, a poisoning attack resulted in an accuracy drop of about 28% by poisoning just 30% of the entire data [176]. If dealing with dosage or treatment management, there may be impacts directly on human lives.

These attacks take place during the training phase (as opposed to inference) of a model. Poisoning attacks as well are not new. In fact, the first examples of poisoning attacks date back to 2004 [177] and 2005 [178], when they were employed to evade spam classifiers.

There are two types of poisoning: against **availability** (investigated in this section) and against **integrity** (also known as "backdoor" attacks, explained in the next section of this essay).

Availability poisoning aims to inject so much poisoned (malicious) data into the system that whatever boundary the underlying model learns, actually becomes useless (due to unacceptable performances). Initially, some work has been done for Bayesian networks [179] and SVMs [180]. More recently, also on neural networks. For example, Steinhardt et al. [181] demonstrated that a 3% of poisoned data into the training set leads to 11% drop in accuracy. Muñoz-González et al. [182] proposed back-gradient approaches for generating poisoned data while Yang et al. [183] used an autoencoder as an attack generator.

Recently, once was discovered transfer learning could be used to train models on limited datasets, attackers discovered they could also transfer the attack together with the rest of the learning. As a practical example, Gu et al. [184] experimented with poisoning attacks on pre-trained models, including a real-world scenario where a US street sign classifier learned to recognize stop signs as speed limits.

In poisoning attacks too, the quantity of knowledge about the model available for the attacker before launching the attack plays an important role in determining the power and possible impacts of the attack itself. Here, the previously presented knowledge classification between white-box and black-box attacks is re-utilized.

Moreover, there is also another important dimension that defines the attacker's capability: **adversarial access**, also known as how deep an attacker can dive into the victim's system. The different degrees of adversarial access are:

- logic corruption (the most dangerous),
- data manipulation,
- data injection,

- transfer learning (the least dangerous).

Logic corruption

Logic corruption consists of the possibility that the attacker can change the algorithm and the way it learns. In other words, this is the case in which the attacker can encode in the model any logic he wants. This attack is essentially equal to the *trojan attack* and, so, will be better detailed in the next dedicated section.

Data manipulation

In data manipulation, the attacker is supposed to have no access to the algorithm itself but to the training set; he can change, add or remove samples from the dataset.

One goal they are able to achieve is *manipulation of labels*. This is useful because, for example, they could add new labels for a subset of the training set and try to optimize the learning task in order to decrease the overall system performance. Availability compromise is relatively simple but installing a backdoor is far more complex. Anyway, this attack requires the attacker to be able to alter a high number of training samples. Biggio et al. [180] demonstrated this attack requires a changing of 40% in the training data by using random labelling flipping; 30% if some heuristics are applied.

Another goal is *input manipulation*. This may lead to a shift in the classification boundary, e.g. a change in the cluster distance [185]. Another option for the perturbation is the addition of an invisible watermark, again used as a "backdoor" into the model or adversarial attack, to gain misclassification and/or model evasion.

Generally, data manipulation is a sophisticated attack, which requires remarkable technical knowledge but also resembles a more realistic threat model. As an example, a realistic scenario can consist in an anti-virus product which continuously collects data for future re-training. In this case, an attacker may submit any type of file but have no control over the labelling process, which may be done either automatically or manually (by a human on the other end). Anyway, if the attacker succeeds in creating a malicious file which gets misclassified, he has just defeated the system.

Data injection

The idea behind data injection is similar to data manipulation; the difference is this is based on simple data addition into the training dataset. As a practical example, Perdisci et al. [186] fooled *Polygraph*, a worm signature generation tool, by inserting perturbations in worm traffic flows.

Transfer learning

The last possible adversarial access is transfer learning. It is the weakest because - as Gu et al. [184] demonstrated - while the network goes through its second training phase, its original weights (including the poisoned ones) get "diluted".

Regarding the quantity of data an attacker has to poison, there is an important consideration. Of course, if the attacker could modify all the training data, he could basically twist the model. This is practically infeasible in most of the cases. According to the research literature, it seems that an attacker is generally able to manipulate a maximum of 20% of the entire training set [187]. Beyond that, the threat model starts to sound unrealistic.

Regarding defences, the situation is very similar to the adversarial attacks. Existing methods do not guarantee robustness in 100% of cases. Following are described all available classes of defences.

Outlier detection

The most common type of defense exploits the concept of outlier detection, based on "data sanitization" and "anomaly detection". In a nutshell, the attacker usually attacks the system by injecting new data whose distribution significantly differs from the one of the current data. So, the defender should be able to detect that [188].

The problem is finding a concrete discriminant for the malicious data. In the easiest case, the poison injected is from a different data distribution and can be easily identified.

In the rest of the cases, the attacker could be able to generate poisoning data which seems very similar to the true data distribution (known as *"inliers"*) but still be able to successfully gain a misclassification.

Another difficult scenario for outlier detection is when the poisoned data has been injected before the filtering rules were created. By definition, in this case, the outliers stop being outliers.

Micromodels

Another useful defence is the use of micromodels. This defence was initially employed to clean training data for network intrusion detectors. A number of classifiers are trained; each classifier is trained on non-overlapping epochs of the training set (and these are the micromodels) and evaluated on the entire training set. By using a majority voting of the micromodels, training instances can be marked as either safe or suspicious. The intuition is that attacks can last very little time and can affect only a few micromodels at a time.

Analysis of impact

Another common type of defence is based on the analysis of the impact of new training samples (which should be added to the model with the re-train process) on the model's accuracy. If this data is poisoned, the model's accuracy on the test set will significantly decrease. So, by doing a "sandboxed" run with the new samples before adding them to the real training set, the defender can spot the attack. Two technical implementations of this class of defenses are *reject on negative impact* (RONI) [189] and *target-aware RONI* (tRONI) [190].

Other measures

Finally, other measures are:

- STRIP [191], already described for adversarial attacks.
- Human in the loop [192]: since poisoned data leads to significant boundary shifts, ask a human (i.e. a security analyst) to analyze the situation.
- TRIM [187]: the model iteratively learns from "trimmed" versions of Mean Squared Errors (MSEs) (applicable only for regression).

Trojan (or backdoor) attacks

In a trojan attack, the attacker's objective is to succeed in modifying the weights of a model while keeping its structure unmodified. The result is that an exploited model works fine on normal samples but predicts the **trojan target label** for specific input samples; these samples are specific infected samples which trigger the trojan. In this case, the victim classifier always predicts the trojan target label when samples provided with the trojan trigger are used.

Trojan attacks are becoming very common in cloud and edge deployments of AI-enabled systems [191, 193].

Gu et al. [194] proposed *BadNets*, an algorithm able to build a backdoor in DNN models by injecting a visible square-like trigger in a fixed location into the training data with a target label. Next, they demonstrated the possible impacts of backdoors in a realistic scenario: they created a U.S. street sign classifier that misclassifies stop signs as speed limits when a special sticker is added to the stop sign; this trojan can persist even if the network is retrained for another task and causes an average drop in accuracy of 25% when the backdoor trigger is detected. Later, Ahmed et al. [195] extended this work by adding support for dynamic trigger patterns and locations.

Liu et al. [196] proposed a backdoor attack called the *Trojan attack*, which can be used to reverse engineer a victim model with the goal of synthesizing training data; this way, access to the original training set is no more required. Once the network has been reversed, to generate a general trojan trigger they retrained the model with external datasets in order to inject in it malicious behaviors. Of course, the malicious behaviours are only activated by inputs which contain the trojan trigger. Their proposal does not require tampering with the original training process. Moreover, it takes just a few minutes to hours. Finally, they do not require the datasets that are used to train the victim model.

Yao et al. [197] proposed a method in which they embed the backdoors directly in "teacher models", thus gaining the possibility to survive the transfer learning process. An additional consideration should be made. *Fine-tuning* (or *incremental learning*) is a technique able to add additional capabilities to an existing model without (directly) employing the original training data. It is based on using the original model as the starting point and training it on the new

data. So, one may object that a trojan attack can be performed by fine-tuning the existing weights of a model: this is wrong and, in practice, it is very difficult to use fine-tuning to perform trojan attacks. This happens because fine-tuning weights just tends to make small changes to the weights of the original model and these small changes are not sufficient to significantly modify the behavior of the model. Assume the input face image of a subject, who is part of the original training data, is tampered with by a trojan: if the model is backdoored exploiting incremental learning, it is very likely that recognizes the image with the genuine label instead of the trojan target label. Technically speaking, this happens because the original values substantially outweigh the injected changes.

The immediate conclusion is that an attacker needs to retrain the model rather than fine-tuning it to successfully conduct the attack. Generally, this is not feasible because model publishers publish their trained models and do not provide the training data. One option may be to derive a set of data that can be used to retrain the model with the trojan. This can be done by inspecting each output node in order to reverse engineer the input that leads to its strong activation. For example, if we take an image classification model, one approach starts with an image generated by averaging all the images from a public dataset to obtain an image for which the model generates very low classification confidence (i.e., 0.1) for the target output. The adversarial input reverse engineering algorithm then tunes the pixel values of the image until a large confidence value (i.e., 1.0) for the target output node - which must be larger than those of the other output nodes - is obtained. As a result, the tuned image can be considered as a replacement of the image in the original training set for the target output node. This process must be iterated for each output node to recreate a complete training set. Of course, a reverse-engineered image may not look like a real target image but, from the viewpoint of the algorithm, it serves the same purpose. In other words, if an algorithm is trained with the original training set or the reverse-engineered input set, the resulting performances are nearly the same.

According to [198], even if the literature went further in improving trojan attacks, most of the defence techniques are designed to make the model robust against the attacks proposed by Gu et al. [194] and Liu et al. [196]. What emerges is that the defender's strategy is based on finding some particular characteristics of an attack; once obtained, the next step is finding a way to detect these characteristics or reverse engineer the process of inserting the trojan and discover information about the trigger shape, size, and location so that it can be detected.

Defences

Overall, making the neural network robust against trojan attacks can be done in two different situations:

- *Before attack*: the most dangerous way for a model to be trojanized is by using online services. So, the most logical consequence would be thinking that one can prevent being attacked by avoiding the use of pre-trained models and transfer learning. Anyway, this is

as easy as inefficient because most users do not have enough time and/or computational resources to train their own model from scratch. Therefore, an alternative solution is to design the model in order to make it robust against attacks. This can be done by making the insertion of trojan/backdoor hard or making the model able to efficiently delete the effects and prevent triggering the trojan if it is inserted into the network. Anyway, there are no explored practical implementations and this can be considered a future challenge.

- *Robustness after attack*: this consists in detecting the trojan when the model is suspected to be under attack. If there is white-box access to the model, compromised neurons or weights can be detected. Nevertheless, there are approaches proposed even if dealing with black-box access to the model. Once the trojan has been detected, the next step is mitigating the attack impacts. Although mitigation techniques are still fundamental to protect the model against the attack, detecting trojan before being infected is a more important task: if a trojan can be preventively detected, one does not need to employ time and energy to apply mitigation techniques. On the other hand, if one is able to detect the trojan type or triggering characteristics, specific mitigation techniques can be chosen which could be more time and cost-efficient.

Following is a brief high-level description of all the defence techniques [198]:

Input anomaly detection (IAD): the defender trains more classifiers with the same number of classes in legitimate data; for each classifier, one specific class is labelled as positive and the others are labelled as negative. The logic now is simple: given an input to these classifiers, if it is legitimate there must be one classifier which classifies this input as positive; otherwise, the input is illegitimate.

Re-training (RT): as the name implies, the core idea is to use only legitimate samples to retrain the model, making it forget the trojans and perform correctly on legitimate inputs; this is done by resetting the weights which contain the trojans with the retraining phase.

Input preprocessing (IP): the basic concept is exploiting an autoencoder model as a validator before the input is fed to the algorithm, in order to detect and stop the trojan trigger. The autoencoder should be trained on legitimate data; during the test, its output should be very similar to the input. If this does not happen, means the input belongs to an illegitimate distribution and must be blocked.

Fine-pruning: as the name states, the idea is to prune the model (potentially backdoored) and then fine-tune it with only genuine data. This works because the pruning defense removes backdoor neurons and fine-tuning fixes the performance decrease resulting from pruning.

Backdoor detection by activation clustering (AC): the authors started from the assumption that some malicious data and other genuine samples receive the same classification (i.e., the same output label) due to some details in the network activation functions: the activation functions of neurons can be classified into two distinct clusters when classifying malicious and clean data; the functions of the clean class, which is not affected when poisoned samples arrive,

shows a homogeneous behaviour. To repair the network there are two possibilities: (i) to remove the malicious data and retrain the model from scratch or, faster, (ii) to re-label the malicious data correctly and continue training the model on these samples.

Neural Cleans (NC): this technique is based on the idea that an infected model requires a much smaller amount of perturbation to misclassify a sample into the target label if compared to other uninfected labels. Using different words, trojan triggers create "shortcuts" towards the classification neurons which contain the target label. These shortcuts can be identified by measuring the minimum amount of necessary modification to lead all inputs from each correct region towards the target region.

Trojan detection via cost of sample classification (CSC): starting from the assumption that the output of a trojanized network has some special characteristics, two subnetworks with the same size and data are trained; these networks represent a decomposition of the original trojanized model: the former is the genuine network and the latter is the trojanized one. Then, once an input with a trigger enters the networks, the output of the trojanized subnetwork must exceed the genuine subnetwork output to successfully trigger the malicious behaviour; this effectively happens because, in the trojan subnet, there exist nodes with abnormal parameters values, which in turn lead to a change in the classification. Since the trojanized nodes always produce costs on a specific output of the SoftMax layer, a relationship between cost and nodes can be identified and used to detect a malicious behaviour: the trojan nodes can be identified by comparing the distribution of the node sensitivity on a layer and the produced costs.

DeepInspect (DI): the idea is very close to NC because stems from the fact that the perturbation required to make an input being classified with the attack target label is smaller than the one in the genuine classes.

Strong Intentional Perturbation (STRIP): the main reason this defence works is thanks to the unbounded perturbation produced by the physical object inserted into the input image to work as the trigger; this is thought to make the attack robust against physical factors such as lighting but, actually, its input-agnosticness represents its weakness. The defence is based on intentionally injecting strong perturbations into the inputs of the network and analyzing the output: if the randomness of the predicted outputs - i.e. the entropy measure - is very low, the input is recognized as trojanized; on the contrary, the entropy of perturbed benign inputs is generally high and this results in the fact that the predictions vary greatly.

Artificial Brain Stimulation (ABS): this defence method consists in changing the neuron's activation function and analysing how the corresponding output changes. If the activation of a target label increases by providing a specific stimulation (i.e. input), the neuron is considered trojanized. Anyway, also benign neurons may show this behaviour and, so, these cases must be further investigated.

Trojan detection via Meta neural analysis (MNTD): following the notions of meta neural analysis, this technique requires a classifier to be trained to predict some specific properties of target neural networks instead of data samples. This classifier is trained by using the outputs

of some shadow models, i.e. trojanized or genuine copies of a model trained on the same task as the target model. The classifier can be trained to address two different tasks: one-class learning (the meta-classifiers are trained with only benign models) and *jumbo learning* (the meta-classifiers are trained with shadow models of different types: trojanized with different types of triggers and genuine with clean data).

Evasion attacks

Different from data poisoning attacks, evasion attacks must be conducted after the model has been trained. The attackers initially have no idea about how to manipulate the data to successfully conduct the attack against the system: attackers need to constantly query the system in a trial and error fashion in order to learn how to manipulate the data passed to the model and obtain a misclassification. Of course, this way of doing creates an overhead on the system and this can be used as a solution to identify suspicious behaviours, which consequently may help save the availability of the system and the power consumption, especially if the system is made of devices with limited energy provision.

There are different kinds of evasion attacks: some try to emulate the properties of normal samples in order to hide intrusions (especially in intrusion detection systems) while others are very similar to targeted adversarial attacks because have the objective of crafting adversarial examples able to gain a misclassification. While in [131] is demonstrated that there have been lots of successes in generating adversarial instances for differentiable models, the same is not true for tree ensembles.

Szegedy et al. [9] show that a lot of machine learning models and neural networks are vulnerable to these attacks: the crafted examples differ very slightly from correctly classified instances but they are still misclassified. Moreover, in many cases, different models trained on the same dataset can be evaded with the same examples (*transferability property*).

[199] shows that classical ML classification algorithms, such as SVM and NN, can be successfully exploited with little knowledge about the system and a handcrafted dataset. A context in which this category of attacks may pose a severe risk is malware detection: as an example, Demetrio et al. [200] proposed a novel attack based on just changing a few bytes in the file header (without injecting any other data) which resulted in forcing *MalConv* - a convolutional neural network for malware detection - into misclassifying the handcrafted input.

Since the main goal of this attack is to avoid the correct classification, there are no specific approaches: this objective can be reached simply by using the same techniques presented for adversarial or trojan attacks.

Model stealing (or model extraction) attacks

As its name implies, the ultimate objective of the attacker is to create a clone or reconstruction of the victim model, re-engineer a black-box model, or understand the structure and the properties

of its training data [201]. Different from the previous two attack strategies, this technique does not need any knowledge about the training data and/or the model properties and structure. An attacker just needs access to the model to receive answers to the submitted queries.

In recent years, MLaaS has become very popular since private entities and small and medium enterprises (SME), which cannot afford industrial hardware, can rent the exact computational power for the exact time they need and pay the corresponding cash amount; this amount is usually very lower than buying the corresponding entire hardware structure.

MLaaS models could be the main target of this attack: spending just a few dollars may be sufficient to create a perfect clone of the victim model deployed and accessed over cloud [202]. Of course, the creation of a private copy of the victim model implies copyright issues; moreover, this exposes the victim model to other attacks with different strategies because the attackers, in that case, would have new information useful for crafting adversarial examples [203].

An important consideration must be highlighted: the various attacks presented are not necessarily independent; a lot of attacks can be launched after a model extraction attack has been successfully completed, since it converts the environment from a black box to a white-box. On the other hand, even if a black-box attack has been completed with successful results, one may still launch a white-box attack.

Model stealing is a very old strategy, dated back to 2005: in [204], Lowd et al. were able to develop an effective algorithm for reverse engineering a spam filter model. They proposed a learning algorithm able to perform quite well and easily exceed the worst-case bounds. The algorithms they used are not designed to be efficient in the number of queries but simple to analyze. Anyway, especially with more in-depth domain knowledge, one can significantly reduce the number of queries to be done.

More recently, Tramèr et al. [205] experimented with the use of a shadow training scheme able to "extract target ML models with near-perfect fidelity for popular ML models"; this scheme works with logistic regression, decision trees and neural networks.

Oh et al. [206] used the idea of building meta-models to extract additional details from a model, e.g. the neural network architecture.

Wang et al. [207] designed an attack useful to steal the hyperparameters of the victim model. These hyperparameters are very useful because are "used to balance the loss function and regularization term in the objective function", which are fundamental for the performance of the final model.

Hua et al. [208] "investigated reverse-engineering attacks on CNN models exploiting information leaks through memory and timing side-channels".

No specific and very focused literature is present regarding defences against this attack. One immediate consideration is that if final users use the model by means of a publicly exposed API service, it is not possible to completely prevent model theft. Anyway, another logical guideline is the defender could make the attack too expensive to implement: one example is using a strict access control policy for the model, putting a limit on daily requests for each user and/or IP

address; also, a financial cost to the request could be added; moreover, another very effective measure could be giving only the output labels and not the correlated confidence levels, like against model inversion attacks (again, if it is necessary to disclose these confidence levels, some form of noise - e.g. a Gaussian noise - could be added to mitigate the effects of this attack).

Anyway, generally, the patterns of the queries made by an attacker are likely to be anomalous compared to the ones of good-willing users [209]: a lot of state-of-the-art anomaly detection tools can be employed as a further mitigation measure.

2.5.3 Explainability in AI-enabled Systems

The explainability aspect is not strictly correlated to security or privacy problems but may help to realize more robust algorithms.

While training a model, what typically happens is that a set of features is fed to an algorithm, which tries to find and learn a hidden pattern which emerges from the data, and returns some predictions. In most cases (excluding the algorithms which naturally provide an explanation, like decision trees and rule-based models), the predictions come without any justification/explanation and the users have no idea of the reasons behind the outcome; these models are also termed as **black-boxes**. Thinking and working towards the explainability of the model means that, besides the prediction/decision, the model should also provide the details that cause the prediction/decision. As a consequence, additional functions/interfaces are required to interpret the causes behind each decision [210].

In the literature, generally "interpretability" and "explainability" are terms used interchangeably. By the way, both are relevant but slightly different: *interpretability* shows the extent to which a cause and effect can be observed within a system, while *explainability* represents the extent of explanation/description of an algorithm to a human.

In lots of fields, there is a number of factors supporting the need for explanation/justification of the choices made by a model. As an example, in smart city applications justification and explanation of a model's outcome are very critical for developing the users' trust in the system, which in turn is used to make critical decisions about their lives, such as whether one can be hired or not (AI-based recruitment), is guilty/involved in a crime or not (i.e., predictive policing), etc... [211].

Guidotti et al. [212] state these justifications can be obtained in two macro-ways: with the help of additional techniques/methods able to describe the potential reasons behind the model's decision, which they named as *"Black-box Explanation"*, or directly designing and developing transparent AI algorithms, like decision trees and rule-based algorithms.

To summarize, explainability should be pursued because:

- An explainable model contributes to building users' trust in the technology, and consequently will speed up its adoption by the industry.

- Explainability is a fundamental characteristic of AI models in some sensitive contexts, like smart cities, healthcare and banking.
- Explainable models are less subject to appeals than other models in decision-making.
- Explainability facilitates the detection of algorithms' biases.

There exist a lot of categorizations and criteria for explainable algorithms; in [210] there is an exhaustive overview. To summarize, there are two main categories, (i) **transparent models**, and (ii) **post-hoc explainability**. The former wraps the methods which naturally provide an explainable outcome while the latter contains the methods used for analyzing the models' behavior after the training. It must be noted that there is the need for a trade-off between performance (e.g., accuracy) and explanation for every algorithm: lower accuracy has been observed for transparent models, such as fuzzy rule-based predictors, comparing to the so-called black-box methods, such as CNNs [213]. Anyway, again, explanation and interpretability are fundamental properties in critical context applications, such as healthcare, smart grids, and predictive policing. As a consequence, there is a particular focus on developing post hoc explainable methods in order to keep a better balance between accuracy and transparency.

Explainability vs adversarial attacks

There are various studies about the relationship between explainability and adversarial attacks [214–216]. As previously mentioned, there is common agreement on the fact that explainability makes models more robust against adversarial attacks; moreover, can facilitate the identification of adversarial (malicious) inputs/samples by explaining why the sample should be considered that way [215]. A very important work in this field is SHAP [217]: Fidel et al. [215] employed it to propose a framework which analyzes the relevance/importance of a feature by assigning an importance value for each specific prediction; this contributes to generating "XAI Signatures" which serve as explanations for the internal layers of a Deep Neural Network (DNN) to distinguish between normal and adversarial inputs.

Dhaliwal et al. [218] exploited a different approach, based on the similarity of gradients to distinguish between normal and adversarial inputs. They stated gradient similarity is useful to show the influence of training samples on test ones and this results in different behaviours for genuine and adversarial samples: this makes it possible to detect adversarial attacks with good performances.

As in some of the previous attacks, the other side of the coin is that the explanations/information produced with these explainability methods could also be used by the attacker to generate more effective adversarial attacks [210].

Anyway, an in-depth analysis of the problem reveals that adversarial attacks can also be used to increase the interpretability of a model: Marino et al. [219] proposed an adversarial AI approach to identify the relevance of each feature involved in the predictions made by an

algorithm; the goal is, given a set of misclassified samples, to find the magnitude of changes in the features required to correctly classify the samples: this measure represents an explanation of the misclassification.

Rahnama et al. [220] used some techniques based on adversarial attacks on a DNN to identify the relevance/importance of the features (of a sample) involved in the prediction, in order to explain the predictions.

Finally, [221] proposed four different types of adversarial attacks and two corresponding defence techniques; and then, explained how explainability contributes to enforcing these defence techniques.

2.5.4 Fairness in AI-enabled Systems

Another aspect not strictly correlated with model privacy and security but with an impact on them is *Fairness*. According to [15], fairness is the most prevalent principle in the current literature.

Algorithmic fairness is at the core of the trust requirements for automated decision-making in sensitive domains, to avoid systemic discrimination against protected groups. Many technical notions of fairness are proposed and many algorithms for enforcing such notions are designed, such as the ones in [222] and [223]. Group fairness measures, such as equalized odds [224], suggest equalizing the model's behaviour across groups that are identified based on a protected attribute (e.g., race or gender). Fairness, however, has a cost on the model's performance, as the best decision rules that satisfy a definition of fairness differ from the optimal decision rules [225].

Chapter 3

Conventional systems work

The previously mentioned approaches to address privacy and/or security in conventional systems present some weaknesses. Among the most critical, most of them:

- Are not able to address simultaneously privacy and security issues in conventional systems
- Cannot provide guidance both for non-technical (like a marketing manager) and technical (like a developer) stakeholders
- Can be used only in one specific phase of the Software Development Life Cycle (SDLC) — in most of cases, just "requirements elicitation"
- Have not been validated on industrial projects

The framework that showed the fewest shortcomings in relation to the previous list is the Privacy-Oriented Software Development (POSD) [74]. In particular, its strengths are:

- POSD considers both privacy and security issues, providing guidelines to developers that can be translated into operational practices, software architectures and software code (so addressing all phases of the SDLC)
- POSD supports both the development of brand-new systems and the re-engineering of existing ones
- POSD can be used in conjunction with the legacy development processes already adopted by an organization

In the following section, the core components of POSD are explained.

3.1 POSD Knowledge Base (PKB)

PKB is made of the 5 Key Elements showed in Fig. 3.1: *Principles of Privacy by Design*, *Privacy Design Strategies*, *Privacy Patterns*, *Vulnerabilities* and *Context*.

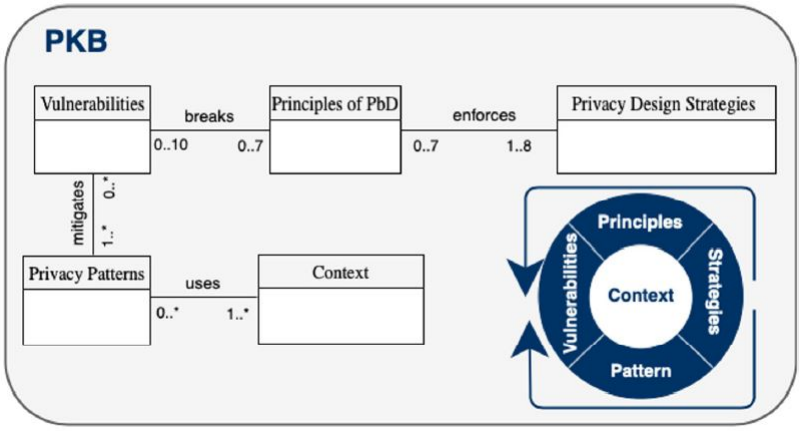


Figure 3.1: The relationship between the Key Elements in PSKB.

With *Principles of Privacy by Design* the authors mean the 7 principles of Privacy by Design (PbD) proposed by Cavoukian [52, 53].

The *Privacy Design Strategies* are the 8 ones proposed by Hoepman [66], grouped into two sub-groups: *Data Oriented Strategies*, called "Privacy-by-Architecture" — which are approaches to prevent and/or mitigate the effects of an attack — and the *Process Oriented Strategies*, also known as "Privacy-by-Policy" — which aim is to inform the user to make decisions according to their needs.

The *Privacy* [226] and *Security Patterns* [227] provide a standard language in the context of protection, documenting and explaining common solutions to common privacy/security problems.

Vulnerabilities are weaknesses which lie within the source code and allow a malicious user to attack the application by undermining data's safety [228].

Finally, *Context* encompasses roles and responsibilities, use cases and scenarios, and functional requirements, determining the flow of data within the system.

All these elements are linked such that the user can understand which principles of Privacy by Design are violated by a vulnerability and which privacy design strategies must be adopted to mitigate it. Operationally, once a strategy has been chosen, a set of Privacy Patterns is associated with it so that privacy requirements can be met.

Furthermore, the knowledge base integrates the results of static code analysis so that any vulnerability identified in the source code of a legacy system is associated with a privacy model and be corrected. Thanks to a connector, the PKB also processes vulnerabilities identified by various static code analysis tools. Each identified pattern can be exported, by means of a translator, into a specific language.

To summarize, PKB provides guidelines to developers at all stages of the software life cycle. These guidelines can be translated into operations by providing the necessary elements for the design of the system architecture and coding. PKB can be used on existing systems and also on systems to be developed. It can be seen as a guided navigation between key elements, choosing

to start with any one of them.

3.2 POSD Enhancements

3.2.1 VIS-PRISE tool

The first drawback identified in POSD framework was the lack of a graphical user interface. This made the framework difficult to use for a non-experienced user. So, I developed the VIS-PRISE tool [229]. It implements the key elements identified in the PKB and their formalization according to the GQM (Goal-Question-Metrics) approach [230]. It has been developed following a Model-View-Controller architecture: the models store in a database the entire PKB; the views visualize the results of the automated static analysis, with all the key elements involved; the controllers receive user requests and process the output.

Following is an example to better understand how the whole tool works, in this case for the backward mode: starting from the static analysis of the source code (SAST) — here done using Fortify SCA — the tool stores information about each security issue in the code and how each one is mapped with the CWE [17] and OWASP Top 10 standards. According to a previously defined *GQM decision table*, based on calculated *metrics* derived from the previous static analysis results, the tool shows the PbD principles to implement in order to mitigate the identified vulnerabilities. In other words, by executing the first *metric model* (Fig. 3.2) —- about PbD violated principles —- is possible to get the list of PbD violated principles; of course, using the other metric models it is possible to obtain information about PbD violated strategies and suggested patterns.

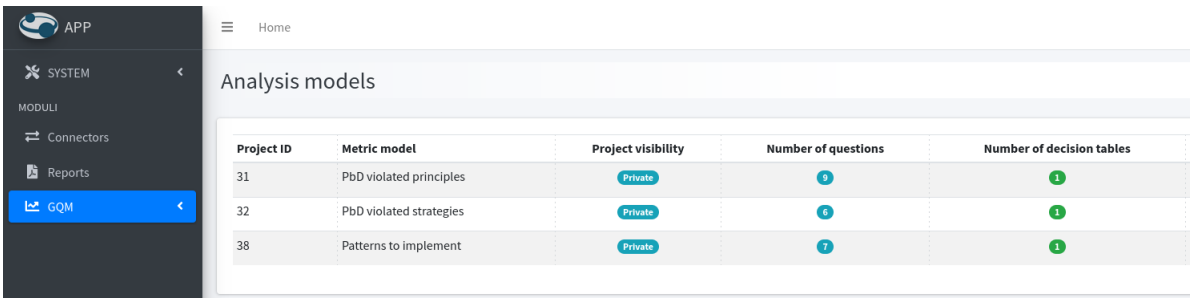


Figure 3.2: VIS-Prise static analysis interface.

The table's execution result is composed of an *interpretation* and the *suggested actions*, as shown in Fig. 3.3. It's also possible to get a PDF version of the obtained results.

3.2.2 MATERIALIST Framework

This chapter describes the work conducted in [1].
Despite privacy is increasingly becoming a desired aspect of software systems — both conventional and AI-enabled ones —, often required by data protection regulations like General Data

Figure 3.3: VIS-Prise results.

Protection Regulation (GDPR), California Consumer Privacy Act (CCPA), Lei Geral de Proteção de Dados (LGPD), different aspects are still limiting and affecting the right implementation of privacy features.

First, designers and engineers often have different ideas about privacy, which is reflected in different approaches and solutions available to implement privacy [65]. Second, data protection regulations such as GDPR only provide legal indications on how systems should be designed and what features should be implemented, but no concrete and technical indications for developers are reported. Third, data protection regulations claim to involve the users [231], but their indications are not user-centric. Fourth, there is often a lack of privacy and security knowledge among developers and engineers [232]. All these aspects contribute to improperly implementing security and privacy features during the software lifecycle, determining vulnerable systems or wrong data protection solutions.

For these reasons, more adequate methodologies are required to ease the process of integrating privacy aspects in the software lifecycle without ambiguities, considering the end-users' point of view and limiting the possible scarce knowledge of the developers and designers on privacy and security.

To address this lack in the current literature, I collaborated with other researchers and we finally added two new components to the PKB: the ISO 9241-210 phases and the GDPR articles.

This research gave birth to the **MATERIALIST** (*Mapping dATa rEgulation softwaRe lIfecycle And vuLnerabilitIeS paTterns*) **framework** [1].

The core result of this framework is still represented by a set of privacy design patterns[1], but here the stakeholders can be guided in using such patterns starting from three different entry points: i) the GDPR articles, ii) the ISO 9241-210 phases and iii) the source code vulnerabilities discovered during static code analysis. The use of these privacy design patterns as the core of the framework aims to solve two of the four issues previously mentioned, specifically (i) different ideas about privacy and (ii) lack of privacy and security knowledge. Indeed, patterns

[1] `https://privacypatterns.org/`

are intrinsically a standardized language for privacy aspects, thus they contribute to having the same idea on privacy. Lack of knowledge is addressed since patterns have been created by privacy experts.

In the MATERIALIST framework, we considered the GDPR instead of other regulations because it is the main privacy regulation in the European Union. On the other side, ISO 9241-210 has been considered since, differently from other processes, it is human-centred and thus contributes to developing user-centric software systems, mitigating the issue that indications provided by data protection regulations are not user-centric.

As previously mentioned, MATERIALIST *traversing* is allowed among any entry points passing through the privacy patterns. For example, a stakeholder may require the list of GDPR articles violated by a specific vulnerability and vice versa; or, a stakeholder may need to know which are the ISO 9241-210 phases that can cause some vulnerabilities and vice versa. This flexible traversing guarantees the possibility to include the privacy design patterns both when the development process is from scratch (forward engineering) and in the case of re-engineering a software system (backward engineering).

In the case of forward engineering, the entry points are the GDPR and the ISO 9241-210. In the first case, to support the stakeholders from the very first phases of the development lifecycle, (e.g., requirements elicitation phase) the patterns suggested help to to comply with the GDPR articles. In the second case, to support the stakeholders along with all the ISO 9241-210 phases, the patterns suggested provide help for each ISO phase.

In the case of backward engineering, the re-engineering software process may start from a source code scan performed by static code analysis tools such as SonarQube and Fortify SCA, that detect common security and privacy source code vulnerabilities (e.g. the ones in the OWASP Top 10). Starting from a specific found vulnerability, the stakeholders can select the privacy design pattern(s) that, if implemented when re-engineering the software, can mitigate such vulnerability.

Regarding the relations within the knowledge base, in MATERIALIST there is not a direct mapping between vulnerabilities and patterns: as detailed in the next subsection, vulnerabilities are mapped with the "Privacy Design Principles", which are already mapped with "Privacy Design Strategies", which are in turn mapped with the "Privacy Design Patterns". At the current stage, the MATERIALIST framework covers the mapping between the GDPR articles and the privacy design patterns.

Mappings between the GDPR and Vulnerabilities

In this section, I provide details specifically related to the forward engineering that starts from the GDPR entry point up to the privacy patterns. This part of the framework addresses one of the current limitations, i.e., GDPR only provides legal indications. To this end, privacy design patterns come in handy as a way to guide the design of systems by proposing a common and reusable solution to common privacy problems. However, privacy patterns alone do not

provide any indication of how to comply with the GDPR, leaving developers with no clear references to the regulation.

To fill this gap and guide stakeholders in selecting the right privacy design patterns according to the GDPR articles they must be compliant with, in this work a mapping between the 72 privacy patterns and the GDPR articles was performed. This mapping has been conducted by two researchers who are experts in both GDPR and privacy patterns. In addition, in order to increase robustness and reduce the possibility of biases, this mapping has been performed starting:

1. from each pattern towards one or more articles of the GDPR;

2. from each article towards the privacy patterns.

In both cases, since the GDPR is composed of 99 articles, most of which are about purely legal and bureaucratic aspects, only the articles related to technical aspects were considered, including the principles (described in Article 5), and the rights of the data subject (described in Articles 15-22).

The two researchers started by performing independently the mapping phase. Each of them spent around 40 hours in this phase. Then, they compared their results and the reliability value was 67%; thus, the researchers discussed the differences and reached a full agreement on the remaining mappings.

Mappings with ISO 9241-210 and Vulnerabilities

Concerning vulnerabilities, the research team extended POSD PKB by updating the already existing mappings between vulnerabilities and privacy design patterns. In particular, the team considered the updated OWASP Top 10 2021 instead of OWASP Top 10 2017 used in the existing mapping. Since MATERIALIST is part of a wider research project, this mapping is still in progress: the team is currently performing the mapping process by associating each vulnerability to one or more privacy patterns. Later, starting from the previously explained mapping between vulnerabilities and principles, principles and strategies, and strategies with patterns, the team derived the mapping. Table 1 in [1] reports an extract of the results of this mapping phase; it shows five OWASP Top 10 2021 vulnerabilities mapped with a number of privacy patterns.

Regarding ISO 9241-210, the team is also carrying out a mapping between the process phases and each PDP. ISO 9241-210 proposes a complementary and iterative process, divided into an initial planning phase and 4 iterative activities (*understand and specify the context of use; specify the user requirements; produce design solutions to meet user requirements; evaluate the designs against requirements*) that are cycled until the solution satisfies all the user requirements. To this end, the mapping the team realized allows to identify, during the development process, when stakeholders should start considering the implementation of a pattern. This is necessary

because security must be considered as a process throughout the whole SDLC. Although one may think that patterns, being practical solutions, may be considered only during the coding phase, their implementation heavily depends on the context and on the requirements of the system. As an example, the *Data Breach Notification* pattern should be considered from the requirements phase as it is mandatory, by the GDPR, that the authorities are notified of the incident within 72 hours. Hence, specific requirements must be set to duly implement this pattern and thus comply with the regulation. On the other hand, a pattern such as *Aggregation gateway*, which deals with more architectural and technical aspects instead, belongs to the design production phase.

In this way, having a clear view on when engineers should start considering a pattern, helps in easing the secure implementation of the system. Moreover, the adoption of a user-centred development process such as ISO 9241-210 further supports the adoption of more usable approaches in security, as usability is frequently overlooked in security and privacy features [233, 234]. This mapping is, however, still a work in progress and will be finalised in future.

Mappings with user stories

Traversing the MATERIALIST framework from any entry point to PDPs can undoubtedly simplify the work of the stakeholders, also avoiding choosing the wrong PDPs due to missing or different knowledge on privacy. However, sometimes this traversing leads to the selection of several patterns, which have to be further refined by the stakeholders if not properly guided. For example, starting from Article 5 of the GDPR, 23 patterns can be used and the selection of the right one(s) is in charge of the stakeholders, which has not furthered elements to guide the choice.

To assist the selection of relevant patterns, the team defined an intermediate layer between the three entry points and the PDPs. This layer consists of a set of common use case scenarios (e.g., *user registration on a website*, *user deletion*) mapped with the PDPs. The team started from a set of user stories proposed by the *Agency for Digital Italy* (AgID)[2], which is the technical agency of the *Presidency of the Council of Ministers*. Such scenarios represent the majority of the situations that can cause privacy and security vulnerabilities.

To perform this map, an analysis of the privacy patterns was made to identify which patterns are involved in each user story and can be implemented to provide a secure and GDPR-compliant procedure (e.g., *secure credentials storage during registration*). As an example, we can consider the users' registration scenario: this is an interactive activity where users are required to input their data, choose a secure password and possibly make some choices regarding how to share their data with the website or any other related third-party services they are registering with (e.g., share the email with a third-party service for marketing purposes). In this context, the designers not only have to design a privacy-aware component but also have to work on the front end to provide an intuitive and clear user interface that can guide users during the whole

[2] `https://www.agid.gov.it/`

registration process and can help in making appropriate decisions, exactly in line with what GDPR asks for.

The PDPs related to this scenario are proposed by the MATERIALIST framework: for example, the *Protection against Tracking* and *Strip Invisible Metadata*. In particular, the following are the suggested PDPs:

PDP	Protection against Tracking
SUMMARY	Do not collect unnecessary cookies, especially if they are useful only in the future and not at the moment
USER STORY	Collecting cookies during registration
NOTES	If it is necessary to collect cookies for system functionality, collect only those strictly necessary and not the optional ones (i.e. technical and profiling ones). For each cookie used, keep a correspondence in the DB between the cookie and the user.

PDP	Strip Invisible Metadata
SUMMARY	At least warn the user of what he is sharing and give him a chance to rectify it
USER STORY	Metadata collection during registration
NOTES	If it is necessary to collect metadata for system functionality, collect only what is strictly necessary and inform the user of all information that can be deduced from the collected metadata.

The stakeholders that start from an entry point thus can further refine the selection of PDPs considering the scenarios they have to cover in their system. As for the ISO 9241-210 and for the vulnerabilities, this is a work in progress.

3.2.3 Empirical Study

This section describes the work conducted in [3].
In an industrial software engineering context, quality is a factor that must be considered from different perspectives. For example, it is often an implicit requirement: stakeholders with little knowledge about software engineering assume that software is inherently reliable, secure, modifiable, adaptable, and so on, and therefore focus their attention on functional aspects and are surprised, if not unwilling, to pay for quality or to compromise on features in exchange for quality. In contrast, software failures, and in particular security vulnerabilities, can cause damage, lead to legal consequences, or have irreversible consequences for the reputation of the software producer.

Moreover, as popular project management frameworks suggest, spending time on quality implies spending less time on implementing features (assuming that the productivity of a team cannot be increased in the short run), or increasing costs (e.g., if we hire more engineers), or prolonging the time-to-market [235].

To mitigate the above-mentioned dilemma, industry has always been interested in reducing quality assurance costs, for example, as suggested by the Toyota Production System [236] and the subsequent Lean Thinking movement [237], through automated quality assurance mechanisms, in Japanese called *Jidoka*, in which "*machines are able to detect the production of a single defective part and immediately stop themselves while asking for help.*" In software production, such automated quality assurance mechanisms are represented by software tools that are able to analyze the produced source code and warn developers of potential issues.

Various software tools to analyze software quality, and in particular code quality, exist. A popular category of these tools is represented by *static* analysis tools that, unlike *dynamic* tools, analyze source code without executing it. Executing source code requires reproducing the hardware context in which the source code is supposed to run, which can be non-trivial. For example, dynamically analyzing the software for an airport baggage handling system, in which custom-made hardware is also involved, is complex since many physical parts need to be simulated truthfully. Often, e.g., to determine performance issues or energy consumption, a dynamic analysis is necessary (e.g., as in [238–240]) but if feasible, analyzing source code statically is often preferred for cost reasons and sufficient to discover a variety of issues.

Therefore, the overall goal of software quality—regardless of how the code is analyzed—is to uncover potential quality problems in the source code. Even if software is developed using Agile methods, to allow changes that may even be late in the process [241], the development team has to consider an exponentially rising cost-of-change curve [242], i.e., it is always better to solve issues early in the process than later. Also from a technical debt point of view, it is in the interest of practitioners to maintain and improve code quality early in the process to reduce the risk of technical bankruptcy [243], i.e. that because of low quality, changes to the software take an unreasonable amount of time or that features cannot be implemented without major redevelopments of large parts of the software.

For the above-mentioned reason, practitioners and researchers are aware of the benefits and need for code quality analysis tools and the adoption is increasing over time [244, 245]. Moreover, the impact, accuracy, and implications of using static analysis tools in a software development workflow are studied [246, 247]. Examples of such studies include comparisons of tools in terms of capability [248], detection agreement, and precision [249]. Other studies compared specific aspects detected by the tools, such as security [250] or concurrency defects [251, 252].

Two very popular static analysis tools are SonarQube[3] [244] and Fortify Static Code Analyzer[4]. SonarQube analyzes code quality based on a customizable quality model and applies the concept of technical debt to inform developers not only about code quality issues but also about the estimated (accumulated) cost of removing those issues.

Not intended as a static analysis tool for code quality in general, but specifically to analyze and test applications for security vulnerabilities, Fortify Static Code Analyzer (Fortify SCA),

[3] `https://www.sonarqube.org`

[4] `https://www.microfocus.com/en-us/cyberres/application-security/static-code-analyzer`

has been named for the 9th time one of the market leaders by Gartner in their application security testing market study, also in 2022 [253]. We found the term "Fortify SCA" or "Fortify Static" 54 times on IEEE Xplore searching within the full text of articles. We conjecture that this lower popularity in academic literature is not because security is less important than code quality but because Fortify SCA is only available with a commercial license, while SonarQube is available Open Source and because the topic of "security vulnerabilities" is more specific than "code quality".

The two tools discussed here—SonarQube and Fortify SCA—have been compared in several other papers, e.g., in [254] where the authors study the challenges in responding to alerts of static analysis tools, in [255], where the authors investigate what developers want and need from program analysis, or in [256] where the authors analyze to identify the limitations of static analysis security testing tools; we conclude from this even more that the research community considers these two products valid tools in the field of static source code analysis and worthwhile comparing their capabilities.

In many contexts, software security is a crucial aspect to consider during the software development process [257] in which developers aim to design the system to be resilient to external attacks [258]. Software vulnerabilities can cause loss of data, privilege escalation, race conditions, and other undesired effects that may affect the source code [259, 260].

The research community has been addressing the problem of vulnerabilities from different points of view, such as the impact on source code [261–263], or developing automated detection techniques [264].

Most of the approaches defined so far are based on source code and/or dynamic analysis [265], symbolic execution [266], and fuzz-testing [267, 268]. Some of them are also implemented within automated tools, such as Coverity Scan[5] or Fortify Static Code Analyzer.

Because of the importance of predicting, identifying, and resolving software vulnerabilities, researchers are also interested in correlating software vulnerabilities with other metrics. For example, Scandariato et al. [269], in study to which extent it is possible to use text mining methods to predict the same vulnerable software components as identified by Fortify SCA.

In order to address the aforementioned issues and to understand if enhancing the code quality also improves software security, I designed and conducted a user study, with the help of other researchers, aimed to investigate whether and to what extent improving software quality could have a positive impact on software security as well. To do so we used as a basis of our investigation two representative tools from both domains: SonarQube for the code quality domain and Fortify SCA for the software security domain. The results of this study have helped us in deciding if the PKB should include both the results of quality-oriented and security-oriented SAST or if security-oriented SAST was sufficient.

The obtained results revealed findings interesting for researchers as well as practitioners: resolving software quality issues using SonarQube has an impact over software security as

[5] `https://scan.coverity.com`.

identified by Fortify SCA, but only to a limited extent. Many security issues still remain in the source code also after improvements. Moreover, some quality characteristics are more likely to be improved with respect to others. Regarding the tools' rules mapping we found new unexpected trends. This study confirms the need for using a security-oriented Static Analysis Tool to enforce software security instead of relying only on a quality-based one.

Goal and Research Questions

The overall goal of this study was to investigate the relationship between refactoring code quality and software security, with the purpose of understanding whether and to what extent improving software quality could have a positive impact on software security as well. The perspective is of researchers and practitioners, both interested in understanding whether these two aspects are related. The former could work on top of our research to build additional knowledge on explaining the reasons behind this relation, while the latter could be interested in applying such a process in their business to increase the efficiency of removing code vulnerabilities. To implement this goal, we defined two Research Questions (RQs).

RQ_1 Do SonarQube rules overlap with Fortify SCA rules?

RQ_2 To what extent does software security improve by improving code quality?

First, we formulated a preliminary research question aiming at understanding if issues in the code raised by SonarQube rules are also considered security issues by Fortify SCA. In particular, considering that SonarQube classifies rules into three categories (Bug, Code Smell, and Vulnerability), we expected to find a perfect match of all the security vulnerability rules detected by SonarQube with those detected by Fortify SCA. However, considering that SonarQube rules are often misclassified [270], we also wanted to verify if the code affected by issues related to rules classified as Bug or Code Smell was also raising security issues from Fortify SCA.

If this research question provided a negative answer, it would have not made sense to address RQ_2

Once ensured that a possible relation between the two investigated aspects existed, we wanted to evaluate the extent to which improving software quality leads to improving software security as well. Thus, we defined the second research question.

Context

The context of this empirical study consisted of projects and static analysis tools. As already stated in the previous section, for the selection of automatic static analysis tools we relied on two tools, based on their popularity in industry and academia: SonarQube for code quality assessment and Fortify SCA for what concerns security assessment.

One could also argue that the selection of the quality-oriented and security-oriented Static Analysis Tools could have been used in this study, since many exist for both aspects. We chose

to rely on SonarQube for quality assessment because it is one of the most used open-source quality-oriented Static Analysis Tools, as various studies demonstrate [271, 272]. Similarly, we selected Fortify SCA to assess projects' security since it is one of the "leaders" in application security testing according to the Gartner 2022 magic quadrants[6]. Additionally, we are aware that changing the group-repository assignments (described below) could have led to a different number of issues in the Fortify SCA final snapshots. Needless to say, also enterprise projects are prone to this risk: quality may depend on the developers' seniority, therefore, this may be considered an acceptable threat. Finally, it is well known that all static analysis tools are affected by false positive and negative identifications, an alert going away may simply mean that the refactoring has made the code too complicated for the tool to follow—which is why it is not reporting an alert anymore—or, conversely, the original alert may have been a false positive which is then hidden. In order to mitigate this problem, as described in section "*Data Collection & Analysis*", a manual inspection has been conducted before and after the refactoring operations.

Regarding the study details, we focused on a set of 23 projects written in Java, each of them satisfying the following criteria:

1. the project is hosted on GitHub (and publicly accessible);
2. its size is greater than 2.5KLOC (i.e., it contains more than 2500 Lines Of Code);
3. it has a SonarQube score worse than "A" for two or more characteristics (i.e., bugs, vulnerabilities, and code smells); and
4. it contains regression tests.

The ratio behind criteria (1) and (2) was to consider only real open-source projects with a non-trivial size. Criterion (3) aimed to discard projects with a good quality score in order to force the participants in our study to spend a significant amount of time to enhance the project's static code quality. Finally, the last criterion allowed us to have sufficient confidence that each applied remediation did not introduce any error in the project logic while enhancing static code quality.

We focused only on Java projects in order to rely on one specific SonarQube quality profile, namely the set of rules checked by the tool. Indeed, by default, SonarQube provides different quality profiles for different programming languages. The same consideration stands for Fortify SCA: there are different sets or rules, each one focused on a different programming language, and just one group of "general rules" (language-agnostic), which was considered in this study.

Table 3.1 gives an overview of the final set of selected projects; their details and URLs of the respective repositories can be found in the replication package (described in Sect. *Replicability* of this chapter).

[6] `https://www.gartner.com/en/documents/4001946`

Table 3.1: Projects' characteristics overview

Name	Description	Size (KLOC)
nbvcxz	Password strength estimator	3.5
CalendarFX	Framework for creating sophisticated calendar user interfaces	33
openwayback	Engine to play back archived websites in the user's browser of the Internet archive Wayback Machine	45
jpmml-sklearn	Library and command-line application for converting Scikit-Learn pipelines to PMML	13
cache2k	In-memory high performance Caching library	24
Fling	Application to beam video files from computers to ChromeCast devices	3.6
scalable-coffee-shop	Demo implementation using event sourcing with an event-driven architecture, Apache Kafka and Java EE	2.6
jnosql	Framework to create Jakarta EE applications with NoSQL	22
PDFrenderer	Library for rendering PDF documents to the screen using Java2D	20
mybatis-3	Framework making it easier to use a relational database with object-oriented applications	23
json-schema-validator	JSON schema validator	8.1
esapi-java-legacy	Web application security control library	19
db-scheduler	Persistent cluster-friendly scheduler	3.5
jackson-databind	General-purpose data-binding functionality and tree-model for the Jackson Data Processor	69
arquillian-core	Component model for integration tests, with dependency injection and container life cycle management	24
javaewah	A compressed alternative to the Java BitSet class	8.2
JSCover	JavaScript Code Coverage Tool that measures line, branch and function coverage	4.1
chromecast-java-api-v2	Java implementation of ChromeCast V2 protocol client	4.6
gchisto	A garbage collection log visualisation tool	8.4
Wikidata-Toolkit	Library to interact with Wikibase	21
initializr	Quickstart generator for Spring projects	16
DroidQuest	A Java recreation of the classic game Robot Odyssey	31
NFE	Electronic Invoice handler written in Java	88

Data Collection & Analysis

In order to address the research questions, we first collected data from the subject repositories and tools and then we analyzed them. To this aim, we followed a two-steps approach. First, to address RQ_1, we performed a manual validation aiming to find correspondences between quality issues raised by SonarQube and security issues output by Fortify SCA. Then, to address RQ_2, we conducted a user study in which we asked participants to perform quality refactoring operations on the subject repositories and then we evaluated the impacts of such refactoring operations on security.

RQ_1– the Manual mapping

The objective of the first RQ was to find a correspondence between quality and security issues. To this aim, we relied on the issues raised by the two aforementioned Static Analysis Tools: SonarQube for quality, and Fortify SCA for security issues.

As our study only focused on Java projects, we considered all the rules included in the

SonarQube default Java quality profile, namely *SonarWay*[7]. This resulted in 627 SonarQube rules.

As for Fortify SCA, we selected the set of rules characterized by two possible values for "*Code Language*": "*Universal*" and "*Java/JSP*"[8], with a total number of 672; the rule pack used was version *2021.2.0.0008*. Then, we compared SonarQube's Java rules and Fortify SCA's ones through a manual validation process. Specifically, for each of the 627 SonarQube rules, two researchers of the group separately read the description and tried to match the rule with one of the Fortify SCA's rules. In case of a disagreement, a third researcher manually verified and took the final decision.

Finally, for each match found, we also annotated the suggested CWE both from SonarQube and Fortify SCA in order to verify the actual existence for a correspondence.

RQ_2– the User Study

To address the second research question, we needed to collect data about the impacts of quality refactoring on software security. Hence, we designed a user study in which we asked participants to perform refactoring operations with the aim solving quality issues raised by SonarQube. Then we compared the issues generated by Fortify SCA before and after the quality refactoring to understand whether and to what extent there was an improvement in terms of security. We ran the study within the Computer Science program at the University of Bari. All participants were third-year undergraduate students who were attending the "*Software Quality*" course. This course is composed by both face-to-face and laboratory lessons, and covered a wide set of topics: all aspects related to software quality (i.e., internal, external, and in-use); ISO standards related to software quality; software quality assessment, monitoring, and improvement processes and strategies [273]; and suggested tools for quality management (e.g., SonarQube), security management (e.g., Fortify SCA) and process control [274].

We initially recruited 128 participants, grouped in 58 groups of 2 or 3 people. At the time of the analysis, only 27 of these groups had completed all their tasks, hence this work discusses only the results of these first 27 groups.

Students were asked to perform their tasks by working in teams. Specifically, each student group performed the following steps on one of the projects reported in Table 3.1:

1. Clone the project repository from GitHub to their local machine;
2. Run the code quality analysis by means of the SonarQube's cloud version (i.e., SonarCloud);
3. Run the code security analysis via Fortify SCA;
4. Annotate the list of issues presented by each tool, comprising, for each issue, file name and line;

[7] `https://docs.sonarqube.org/latest/instance-administration/quality-profiles`
[8] `https://vulncat.fortify.com/en/weakness?codelang=Java%2fJSP%3bUniversal&q=`

5. Incrementally solve the issues output by SonarCloud, following the SCRUM methodology: in each "action plan" (equivalent to a SPRINT), a subgroup of issues was selected and solved. The quality refactoring operations were performed manually by the students, according to the issue and the correspondent solution provided by SonarCloud. At the end of each "action plan", another SonarCloud scan was run in order to check that each issue was actually removed; and

6. Run the code security analysis again via Fortify SCA once a quality target was reached.

The quality target varied according to the number of team components and the project's complexity which in most cases was: 0 bugs, 0 vulnerabilities, 0 code smells and a technical debt of 0 days. The exact numbers for all the projects are reported in the online appendix, see [275].

One observation needs to be made: in this study we needed to run Fortify SCA both before and after the refactoring operations. This was necessary because we needed a quantitative yardstick to compare how the project's security was affected by the quality refactoring.
We analyzed the resulting data by comparing the total number of occurring Fortify SCA issues before and after the quality refactoring was performed, both in absolute terms and percentages. In case the number of Fortify SCA issues changed after the refactoring operations, a manual validation process was performed: one of the researchers manually compared the affected source code with its own version before the refactoring; if the vulnerability was effectively removed (according to SonarQube suggestions) and Fortify SCA did not signal the issue anymore in the refactored source code, the refactoring was considered as valid and taken into consideration in the metrics count. This way, we did our best to exclude the presence of false negatives (i.e. genuine vulnerabilities not detected due to a random Fortify SCA check failure).
In order to mitigate false positive occurrences (i.e. pieces of code flagged as vulnerable but effectively not vulnerable), additional measures were taken: before and after the refactoring, an average of 45% of the total remaining issues - chosen by random sampling - was manually inspected for each project; if a false positive issue was discovered, it was suppressed (removed from the issues count on Fortify SCA).

After this filtering, we then computed three additional metrics, based on the rules mapping performed in RQ_1. In particular, we computed the "*match count*" as the number of issues identified by both SonarQube and Fortify SCA in the initial snapshot; only rules with a match in the mapping performed in the context of RQ_1 were taken into consideration. Moreover, we measured the "*solved count*" as the number of matching issues that were spotted by Fortify SCA in the initial snapshot but were removed in the final snapshot, i.e., the number of Fortify SCA issues solved following to the quality refactoring operations. Again, only rules with a match in the mapping performed in the context of RQ_1 were taken into consideration. We calculated the "*solved ratio*" as the ratio between "*solved count*" and "*match count*".

Finally, we used statistical tests to confirm the variations in performance amongst the Fortify SCA issues distribution before and after the refactoring. To this aim, we relied on the paired

Wilcoxon test [276]. At *alpha*=0.05, the results were supposed to be statistically significant. We also exploited Cliff's Delta (or d), a non-parametric effect size measure [277] for ordinal data, to assess the magnitude of the measured differences. To interpret the effect size values, we used well-established guidelines: negligible for $|d| < 0.10$, small for $|d| < 0.33$, medium for $0.33 \leq |d| < 0.474$, and large for $|d| \geq 0.474$ [277].

Replicability

To allow other researchers to replicate our study, we have published the complete raw data in the replication package [275].

Results

RQ_1: Do SonarQube rules overlap with Fortify SCA rules?
Table 3.2 reports the overall results regarding rule matches. As we can see, most of the SonarQube rules are related to design (i.e., code smells) or potential bugs. However, as one could expect, just a few portions of these rules find a match with Fortify SCA ones whose main focus is on security. Differently, for those rules aiming to identify vulnerabilities or security hotspots, we found a strong overlap between the two considered tools, namely 81% for vulnerabilities and 72% for security hotspots.

Table 3.2: Aggregated analysis regarding the rules mapping. For each SonarQube category, the table reports the number of SonarQube rules (#rules), the number of rules matching with Fortify SCA rules (#matches), and the percentage of rules matching with Fortify SCA rules (%matches).

SQ categories	**# rules**	**Fortify SCA**	
		# matches	**% matches**
Bug	149	26	17%
Vulnerability	53	43	81%
Security Hotspot	36	26	72%
Code Smell	389	12	3%
Total	627	107	17%

The results suggest that SonarQube can contribute to a considerable extent (17%) in identifying and managing security-related issues. If we look at the other side of the coin, the results suggest that around 16% of Fortify SCA issues have a correspondence in SonarQube (i.e., 107 out of 672). Therefore, even if the results suggest that it is not possible to have significant coverage of Fortify SCA's rules, it seems that SonarQube could contribute by discovering a good percentage of issues of certain types. In the next section, related to the second research question, we analyze the magnitude of such a contribution.

RQ_2: To what extent does software security improve by improving code quality?

Table 3.3 reports the number of Fortify SCA issues, grouped by severity levels, before and after the refactoring. Overall, by applying refactoring only relying on the output of SonarQube,

Table 3.3: Number of Fortify SCA issues before and after the refactoring, grouped by severity.

Severities	initial snaphot	final snaphot	% solved
Critical	410	368	10%
High	3089	2651	14%
Medium	114	108	5%
Low	12052	9487	21%
Total	15665	12614	19%

Table 3.4: Overall results regarding how projects' security is impacted by refactoring based on SonarQube's suggestions.

Group	Fortify SCA Total vulnerabilities (initial snapshot)	Fortify SCA Total vulnerabilities (final snapshot)	% Fortify SCA issues removed	Post-Refactoring situation	Match Count	Solved Count
G1	125	125	0	Equal	0	0
G2	317	296	7	Improved	64	64
G3	2651	2521	5	Improved	21	16
G4	37	24	35	Improved	2	2
G5	421	431	-2	Worsened	30	5
G6	251	272	-8	Worsened	6	6
G7	44	84	-91	Worsened	2	0
G8	149	154	-3	Worsened	0	0
G9	390	330	15	Improved	29	29
G10	613	661	0	Improved	10	5
G11	66	56	15	Improved	1	1
G12	1116	805	28	Improved	47	47
G13	1116	815	27	Improved	47	47
G14	119	69	42	Improved	0	0
G15	1047	802	23	Improved	69	57
G16	251	220	12	Improved	7	7
G17	610	575	6	Improved	43	13
G18	1508	276	82	Improved	19	19
G19	11	33	-200	Worsened	0	0
G20	227	170	25	Improved	0	0
G21	31	17	45	Improved	0	0
G22	87	115	-32	Worsened	0	0
G23	269	266	1	Improved	7	6
G24	215	218	-1	Worsened	0	0
G25	1921	1491	22	Improved	85	85
G26	149	150	-1	Worsened	0	0
G27	1494	1451	3	Improved	66	36

leads to an improvement of 19% in terms of Fortify SCA solved issues. Inspecting the results by severities, the results report that SonarQube can help identify, and then solve, 10% of critical issues, 14% of high-severity issues, 5% of medium-severity issues, and 21% of low-severity issues.

Table 3.4 reports the overall results regarding how projects' security was affected by implementing SonarQube's suggestions. Observing the table, in 18 out of the 27 cases projects got enhanced in security terms, 1 remained stable while 8 got a deterioration.

Columns "*match count*" and "*solved count*" of the table report the number of matching

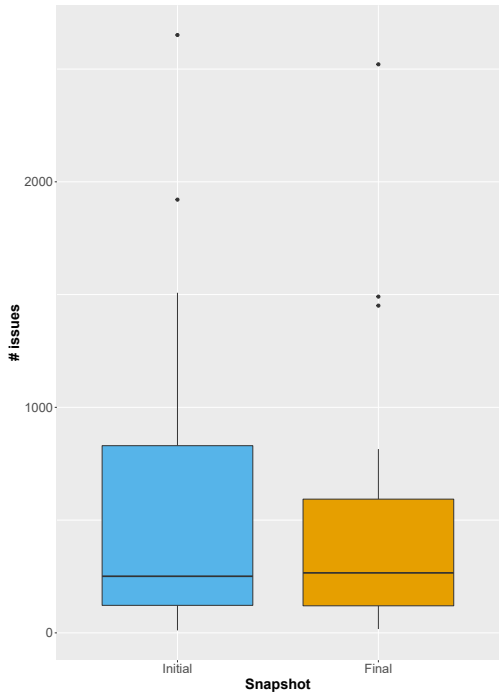


Figure 3.4: Boxplots reporting the distribution of Fortify SCA issues over project in the initial (before refactoring) and the final (after refactoring) snapshots.

Table 3.5: Results for the paired Wilcoxon's signed rank test for statistical significance and the Cohen's "d" effect size.

W	p-value	significance	d	Magnitude
283.5	0.006	***	0.416	Medium

Fortify SCA issues and solved matching Fortify SCA issues for each project, respectively.

As we can see, the maximum obtained enhancement is equal to 82% (for group G18) which corresponds to the solving of 1232 issues. Anyway, this happened just one time: in most of the cases, the obtained enhancement is between 0 and 45%, leaving a significant amount of issues in the source code.

On the other hand, when the number of Fortify SCA issues increases after refactoring (i.e., there is a decrease in security), there are a couple of concerning examples reporting a security decrease of 91% (G7) or even 200% (G19). This could be related to more delicate refactoring operations that lead to introducing new security issues.

Overall Results

Overall, in most of cases, the vast majority of matching Fortify SCA issues are solved after applying the refactoring on top of SonarQube warnings. Moreover, we can observe that in 6 out of the 9 cases (i.e., 67%) in which there are no matching issues between Sonar and Fortify SCA, performing refactoring operations based on SonarQube warnings does not bring improvements in terms of security.

Table 3.6: Co-occurring rules between SonarQube and Fortify SCA

SonarQube rule	Fortify SCA rule	# occurences
Delivering code in production with debug features activated is security-sensitive	System Information Leak	232
Unused method parameters should be removed	Dead Code: Unused Method	22
Null pointers should not be dereferenced	Redundant Null Check	14
Unused method parameters should be removed	Poor Error Handling: Overly Broad Throws	10
Standard outputs should not be used directly to log anything	Privacy Violation	9
Mutable fields should not be "public static"	Password Management: Hardcoded Password	8
Null pointers should not be dereferenced	Insecure Randomness	3
Null pointers should not be dereferenced	J2EE Bad Practices: Threads	1
Null pointers should not be dereferenced	Denial of Service: Parse Double	1
Silly equality checks should not be made	Code Correctness: Class Does Not Implement equals	1
Return values should not be ignored if they contain the operation status code	System Information Leak: Internal	1

Figure 3.4 reports boxplots comparing the distributions of Fortify SCA issues over projects before and after the refactoring is performed. While the median values are almost identical, we can observe that the distribution in the final snapshot concentrates more towards lower values, thus indicating an overall improvement. To better investigate the significance of such a result, Table 3.5 reports the results of the paired Wilcoxon's signed rank test, as well as the Cohen's "d" effect size statistics. Results indicate that there is a significant improvement with respect to the number of security issues spotted by Fortify SCA, with a "medium" effect size.

Therefore, we can state that adopting SonarQube and refactoring the source code according to the warnings it generates, most likely leads to a significant improvement in terms of security aspects. However, it is still not possible to completely replace security-specific automatic static analysis tools (i.e., Fortify SCA in our case) since they can provide a more comprehensive overview of security aspects, identifying issues that cannot be identified in other ways.

Discussion

The achieved results revealed a number of insights that lead to implications for the software engineering community.

On the correspondence between SonarQube and Fortify SCA rules. We identified co-occurring rules that were triggered by both SonarQube and Fortify SCA on the same source code file line. Please note that these co-occurrences were not expected since we did not include them in our initial mapping ($\mathbf{RQ}_1$). This was due to the fact that corresponding rules descriptions seem not to be similar enough. The co-occurring rules are presented in Table 3.6. As we can see, some of these trends are very rare to occur (just one occurrence for some of them), thus not necessarily indicating a real mapping between the two involved rules. However, in other cases, especially for the first row, it seems that there is a real association between the

rules of the two tools.

By focusing on the descriptions of the two rules having 223 matching occurrences (i.e., row 1 of Table 3.6), even if both rules are abstract and do not address a specific problem (rather a general condition), we can see both have the word "*debug*" and one code example in common: the calling to *printStackTrace()* method over an *Exception* object; this seems to be fair and makes sense: so, this may be the case in which both the authors did not notice a valid association while carrying out the mapping.

In the remaining cases, the rules' descriptions do not show anything in common and, as such, may be random (or noisy) findings.

> 🔑 Even if some SonarQube and Fortify SCA rules are not directly associated, a high number of co-occurrences between them exist, thus indicating that further and larger investigations are needed to determine the actual set of corresponding rules.

Better together: One tool is not enough. According to the results obtained in our study, and specifically for $\mathbf{RQ}_1$, SonarQube is able to cover a small percentage (i.e., 16%) of the rules included in Fortify SCA. However, the most interesting finding has been obtained by looking at the severity of the overlapping rules. SonarQube allows to solve security issues classified with higher severity (High and Critical). Even if the improvement is not so significant, SonarQube can be adopted as an initial screening for security healthiness.

In light of these considerations, our results represent a call for further investigation regarding the role of static analysis tools for security issues detection. SonarQube and Fortify SCA classify similar rules differently or provide different classifications. It should be interesting to evaluate if the same observed trend can be recoverable with other static analysis tools, such as *Coverity Scan*.

> 🔑 SonarQube alone is not enough to provide significant support in discovering and removing security issues. However, it could be exploited to have a first preliminary overview of such issues. As a consequence, PKB must include the results of both SonarQube and Fortify static code analyses.

Chapter 4

AI-based systems

One of the primary goals of this thesis work was to propose guidelines and tools to support the development process of both conventional and AI-enabled systems in a company. To do this, I first started studying the state of the art regarding conventional systems, with the goal of discovering common practices and lessons learnt to take into consideration for AI-enabled systems secure development. Later, a similar preliminary research was done to find, and eventually enhance, the best framework for AI-enabled systems.

4.1 AI-enabled systems rapid literature review

This section describes the work conducted in [4].
Before explaining the work and results of this rapid literature review, here are presented some preliminary definitions to better comprehend the guiding concepts around this research.

4.1.1 Trustworthy (or Responsible) AI Principles

National and international organizations have created ad-hoc expert groups on AI to address the risks connected with the development of AI, frequently with the task of generating policy documents. These organizations include, among others, the High-Level Expert Group on Artificial Intelligence established by the European Commission[1], the UNESCO Ad Hoc Expert Group (AHEG) for the Recommendation on the Ethics of Artificial Intelligence[2], the Advisory Council on the Ethical Use of Artificial Intelligence and Data in Singapore[3], the NASA Artificial Intelligence Group[4] and the UK AI Council[5], just to cite a few.

[1] https://digital-strategy.ec.europa.eu/en/policies/expert-group-ai
[2] https://www.unesco.org/en/artificial-intelligence/recommendation-ethics
[3] https://www.cms-holbornasia.law/en/sgh/publication/singapore-to-form-advisory-council-for-ethical-use-of-ai
[4] https://ai.jpl.nasa.gov/
[5] https://www.gov.uk/government/groups/ai-council

These committees have been appointed to produce reports and guidelines about RAI. Similar initiatives are being made in the commercial sector, particularly by businesses that depend on AI. Businesses like Sony[6] and Meta[7] made their AI policies and principles available to the public. At the same time, professional organizations and no-profit groups like UNI Global Union[8] and the Internet Society[9] have all released statements and recommendations.

The significant efforts of such an ample group of stakeholders to develop RAI principles and policies not only show the need for ethical guidance but also point out their keen interest in reshaping AI ethics to suit their individual priorities [278]. Notably, the private sector's participation in the field of AI ethics has been questioned since it may be using high-level soft policy as a portmanteau to either make a social issue technical [278] or avoid regulation altogether [15, 279].

However, many research works highlighted how these proposals often diverged, giving different definitions, resulting in the problem known as *principle proliferation* [280]. Consequently, several in-depth investigations have been conducted, such as the one by Jobin et al. [15], who found a global convergence around five ethical principles: *transparency*, *justice and fairness*, *non-maleficence*, *responsibility*, and *privacy*. Nonetheless, further in-depth thematic analysis revealed notable semantic and conceptual divergences in interpreting these principles and in the particular recommendations or areas of concern drawn from each of them.

Definitions of AI principles

As highlighted in the previous section, there are several uncertainties and nuances around the definition of the principles that mainly characterize Responsible AI, as well as, about the definition of RAI itself. Indeed, sometimes it is referred to as Trustworthy or Ethical AI. In this thesis work, I address the problem of *principle proliferation* deciding to focus on a specific subset of those characterizing RAI, in particular, the four principles identified by Jobin et al. [15] with the exclusion of *responsibility* as this concept is rarely defined in a clear manner.

Moreover, to give an authoritative and clear definition for each principle, I decided to use the ones provided by the High-Level Expert Group on Artificial Intelligence established by the European Commission[10] in their *Ethics guidelines for trustworthy AI* [281].

In the following, I report the selected definitions for each principle. I matched the principles in [281] with the ones that emerged in [15], and when the naming convention is not exactly the same I highlighted that in parenthesis. I also mapped principles to system requirements.

[6] https://www.sony.com/en/SonyInfo/sony_ai/responsible_ai.html

[7] https://ai.facebook.com/blog/facebooks-five-pillars-of-responsible-ai/

[8] http://www.thefutureworldofwork.org/media/35420/uni_ethical_ai.pdf

[9] https://www.internetsociety.org/resources/doc/2017/artificial-intelligence-and-machine-learning-policy-paper/

[10] https://digital-strategy.ec.europa.eu/en/policies/expert-group-ai

Transparency

This requirement is closely linked with the concept of *explicability* [...] Explainability concerns the ability to explain both the technical processes of an AI system and the related human decisions (e.g. application areas of a system).
Technical explainability requires that the decisions made by an AI system can be understood and traced by human beings. [...]. [The] explanation should be timely and adapted to the expertise of the stakeholder concerned (e.g. layperson, regulator or researcher). [...]

This principle includes and can be directly linked with system requirements such as *traceability* and *explainability*.

Diversity and non-discrimination and fairness (Justice and fairness)

In order to achieve Trustworthy AI, [one] must enable inclusion and diversity throughout the entire AI system's life cycle. [...] this also entails ensuring equal access through inclusive design processes as well as equal treatment. [...] Bias [derives from] data sets used by AI systems (both for training and operation) [because these] may suffer from the inclusion of inadvertent historic bias, incompleteness and bad governance models. The continuation of such biases could lead to unintended (in)direct prejudice and discrimination against certain groups or people, potentially exacerbating prejudice and marginalisation. [...]

This principle can be directly mapped with system requirements such as *avoidance of unfair bias*, *accessibility and universal design* and *include stakeholder participation*.

Technical robustness and safety (Non-maleficence)

[...] Technical robustness requires that AI systems are developed with a preventative approach to risks and in a manner such that they reliably behave as intended while minimising unintentional and unexpected harm, and preventing unacceptable harm. This should also apply to potential changes in their operating environment or the presence of other agents (human and artificial) that may interact with the system in an adversarial manner. In addition, the physical and mental integrity of humans should be ensured.

This principle can be directly mapped with system requirements such as *resilience to attack and security*, *fallback plan* and *general safety, accuracy, reliability*, and *reproducibility*.

Privacy and data governance (Privacy)

[...] Privacy [is] a fundamental right particularly affected by AI systems. Prevention of harm to privacy also necessitates adequate data governance that covers the quality and integrity of the data used, its relevance in light of the domain in which the AI systems will be deployed, its access protocols and the capability to process data in a manner that protects privacy.

This principle can be directly mapped with system requirements such as *including respect for privacy*, *quality and integrity of data* and *quality and integrity of access to data*.

Frameworks

In this thesis work, I focus on frameworks that implement the above-mentioned RAI ethical principles.

The concept of *framework* is far well-known in the Software Engineering (SE) field. Already in 1997, Johnson et al. [282] referred to frameworks as "*an object-oriented reuse technique*" or "*the skeleton of an application that can be customized by an application developer*". These are not conflicting definitions; the first describes the structure of a framework while the second describes its purpose.

Shifting the focus from SE to a more general context, frameworks are a form of design reuse. Frameworks can be considered a collection of suggestions, guidelines and tools to be followed in order to create a product compliant with a defined standard.

Study Design

Rapid Reviews (RRs) have emerged as a streamlined approach for synthesizing evidence quickly, initially intending to help decision-makers in health care to respond promptly to urgent and emerging needs [283]. Rapid reviews simplify systematic review methods by focusing on the literature search while still aiming to produce valid conclusions [284].

To perform this rapid review, I followed the protocol proposed in [285], and I complemented the Rapid Review process with the strategies presented in [286] for performing systematic literature reviews. The following subsections describe in detail the study design and its execution.

Planning the review

The rapid literature review presented in this work was carried out through the following steps:

1. **Goal and Research questions**: the goal and the correlated research questions were identified to guide the literature review;

2. **Search strategy**: defining the strategy to collect previous works published in the literature, including research databases and query strings;

3. **Eligibility criteria definition**: the criteria used to filter the collected studies have been defined;

4. **Data extraction**: defining how relevant data were extracted to help answer the research questions;

5. **Data synthesis**: defining how to organize extracted relevant data to answer the research questions.

Goal and Research Questions

The goal of this review is to gather, organize, and analyze the RAI frameworks proposed in the literature by public and private institutions to investigate, first, how comprehensive they are in terms of principles addressed and SDLC phases covered. Then, I focus specifically on frameworks offering tools that can be used in the implementation and auditing phase of the decision systems offering practical guidance to practitioners.

Based on this goal, I defined the following research questions:

- **RQ1**: What are the Responsible AI frameworks proposed in the literature?
- **RQ2**: How much do these frameworks address the various RAI principles?
- **RQ3**: Do these frameworks provide recommendations for each phase of the Software Development Life Cycle (SDLC)?
- **RQ4**: Is there a supporting tool for each proposed framework?

I recall that I decided to focus this research on the most cited RAI principles [15]:

Transparency, **Diversity & Non-discrimination & Fairness**, **Technical Robustness & Safety**, and **Privacy & Data Governance**.

Search strategy

As recommended in [285], to abbreviate the search for primary studies and conduct the rapid review within the available time, I used only **Scopus**[11] and **Google Scholar**[12] as white literature search engines.

To improve the search string, I conducted pilot searches and I excluded those keywords that did not yield coherent search results (e.g., the *knowledge base* keyword). After some trials, I found the search string presented in the following box that returned several relevant papers.

[11] `https://www.scopus.com/`
[12] `https://scholar.google.com/`

Search String

(ai OR (artificial AND intelligence)) AND framework AND (trustworthy OR ethical OR governance OR responsible)

The selection string was run on 15 November 2022 at 12:40 and initially produced 1875 document results on Scopus. Then, some filters were applied: I selected only "Computer Science", "Social Sciences", "Engineering" and "Mathematics" subject areas and selected only documents written in English or Italian; these filters reduced the total amount of results to 1489. After a few days, the same string was run on Google Scholar - filtering the results by "Review articles" type - and produced 91200 results.

The classification process of the documents collected by Google Scholar stopped after 20 pages because, starting from the 21st, the documents did not include all the keywords contained in the search string and/or were not coherent with the research objective; a total of 200 documents collected by Google Scholar were analyzed.

Later, to further enhance the quality of this research, I moved to analyze grey-literature sources. First of all, **Algorithm Watch** AI Ethics Guidelines Global Inventory[13] database was queried to identify the business companies which proposed RAI resources. Here, three types of resources were discarded:

- The ones whose link was no more working and the same resource could be not found even searching on Google.com search engine;
- Unofficial documents, e.g. unofficial translations;
- Documents not written in English or Italian.

In this stage, for each resource found on Algorithm Watch, an in-depth investigation has been conducted on the proponent entity's website to discover useful resources (and frameworks) not indexed on Algorithm Watch AI Ethics Guidelines Global Inventory.

Then, the **OECD database**[14] was queried as well. Here, some filters were applied:

- *Procedural* Category, which (according to the description) is the most similar to the concept of "framework";
- *Published document* or *implemented in multiple projects* as Tool Readiness;
- *Fairness, Privacy & data governance, Robustness & digital security, Transparency & explainability* as Object.

Finally, a keyword-based search on Google was performed in private-browsing mode, after logging out from personal accounts and erasing all web cookies and history [287]. The search was

[13] `https://algorithmwatch.org/en/ai-ethics-guidelines-global-inventory/`
[14] `https://oecd.ai/en/catalogue/tools`

performed using the same search string used on Scopus and Scholar. This last search initially produced 21,100,000 results but, after manually inspecting the results contained in the first 17 pages—for a total amount of 168 resources—, the search engine hid subsequent results since they were very similar to the previously inspected ones. From the Google search results, I also discarded all the duplicates of the documents already retrieved from other data sources.

All the documents obtained with this search strategy were surveyed using a 3-stages information classification process. In the first stage, only the title and keywords of the collected articles were read. In the second stage, I analyzed the abstract of each article while in the third stage I read the complete article. All these stages were conducted separately and in blind-view way by me and then by another researcher. In case of a disagreement, a third researcher manually verified and took the final decision.

Eligibility criteria definition

The selection procedure was based on the following criteria:

1. The resource must be in English or Italian;
2. The resource must be in the context of Responsible AI frameworks;
3. The resource must address at least one of the chosen principles (see Section 4.1.1);
4. The resource must provide answers to at least one of the rapid review's research questions.

The full text of every collected resource was validated against these criteria. The complete results list can be seen in the online appendix [288].

Data Extraction

In this step, I extracted all relevant data that could help answer any of the research questions. The extraction process was performed by me and another researcher and conflicts were solved by a third researcher in a blind-view way. I used a worksheet to tabulate and organize data [288].

Results

The RR was performed part-time from 15 November 2022 until 12 December 2022, conducting the procedure previously described.

Table 4.1 summarises the number of relevant results obtained in each sub-phase. The search in Google Scholar did not produce any useful results. Summing up the resources selected from the identified data sources in the last step, I ended up with 148 unique resources (without duplicates).

Table 4.1: Amount of documents collected grouped by research phase.

Data Source	Resources retrieved	Resources analyzed	Resource selected
Scopus	1875	1489	20
Google Scholar	91200	200	0
Algorithm Watch	167	167	80
OECD DB	356	70	38
Google Search	21,100,00	168	10

Table 4.2: Excerpt of the whole data classified by RAI principles taken into consideration.

Entity name	Tool Name	Diversity & Non-discrimination & Fairness	Privacy & Data Governance	Technical Robustness & Safety	Transparency
COMPANIES					
Meta	Facebook's five pillars of Responsible AI	Yes	Yes	Yes	Yes
UNIVERSITIES					
University of Texas at Austin	CERTIFAI	Yes	No	Yes	Yes
NO-PROFIT ORG / COMMUNITIES / GOVERNMENT ENTITIES					
NIST	AI Risk Management Framework	Yes	Yes	Yes	Yes

Table 4.3: Excerpt of the whole data classified by SDLC phase addressed.

Entity name	Requirements Elicitation	Design	Development	Testing	Deployment
COMPANIES					
Meta	Yes	No	No	No	No
UNIVERSITIES					
University of Texas at Austin (CERTIFAI)	No	Yes	Yes	Yes	No
NO-PROFIT ORG / COMMUNITIES / PUBLIC ENTITIES					
NIST	Yes	Yes	No	No	No

Data synthesis

Data were synthesized and several statistics were computed to answer the research questions (see the online appendix available at [288]). Tables 4.2 and 4.3 show an illustrative excerpt of the whole data collected.

RQ1. *What are the Responsible AI frameworks proposed in the literature?*

All the retrieved frameworks have been classified w.r.t. the **type of proposing institution** using three categories: COMPANIES, UNIVERSITIES, and NO-PROFIT ORG / COMMUNI-

TIES / PUBLIC ENTITIES (NPG/COMM/PE). Moreover, if an entity proposed two (or more) frameworks, it was counted only once, as I am interested in the distribution of the proposals by entity type.

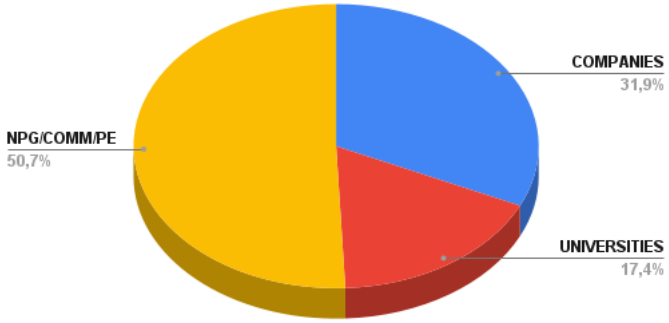


Figure 4.1: Distribution by type of proposing institution.

As can be seen in Fig. 4.1, what immediately emerges is that most of the filtered frameworks are proposed by NPG/COMM/PE (50.7%, 70/138), followed by lucrative COMPANIES (31.9%, 44/138) and then UNIVERSITIES (17.4%, 24/138).

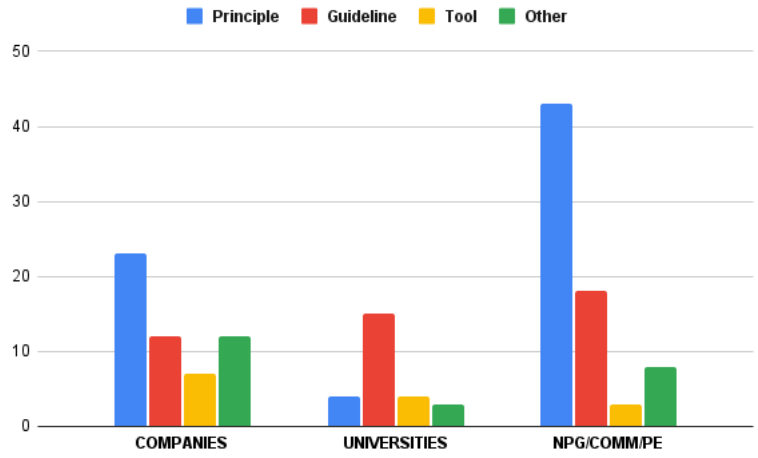


Figure 4.2: Distribution of frameworks by category.

Furthermore, the frameworks have been classified into four categories according to their characteristics:

- **Principle (P)**: if they highlight only abstract ethical principles or moral values;
- **Guideline (G)**: if they are concrete guidelines, quickly translatable into design constraints or choices;
- **Tool (T)**: if they are able to verify the compliance towards one or more principles and/or support practitioners in the implementation of principles or guidelines;

- **Other (O)**: if a resource cannot be classified into any of these categories - e.g. a list of possible attacks against an AI algorithm or a list of several questions to check in the design phase.

In Fig. 4.2 the distribution of frameworks by their category and grouped by proposing institution is depicted[15].

It is noteworthy that COMPANIES and NPG/COMM/PE mostly proposed **Principles** (respectively 42.59%, and 59.72%); differently from this trend, UNIVERSITIES primarily proposed **Guidelines** (57.69%). Furthermore, w.r.t. the number of resources proposed, the percentage of **Tools** proposed by COMPANIES is 12.96% and 15.38% for UNIVERSITIES, while this number is very low for NPG/COMM/PE (4.17%).

RQ2. *How much do these frameworks address the various RAI principles?*

In this review, I am mainly interested in analyzing frameworks that give practical support to all stakeholders involved in the development and deployment of AI applications. For this reason, to answer RQ2, I discarded, while computing the statistics, frameworks in the category *Principle*, which cover only ethical values and that do not give any workable advice. Furthermore, here I counted the frameworks and not the entities: if an entity proposed two (or more) frameworks, it was counted twice (or more).

Finally, before diving into the statistics, I point out that here I consider a principle as "covered" even if it is only "partially" covered, i.e. if not every aspect related to that principle was addressed. For instance, I consider frameworks that deal with privacy but only in terms of how data is acquired from users and stored, without dealing with the new "privacy attacks" which can be conducted against an AI algorithm, like "model inversion" attack [122].

As shown in Fig. 4.3, in the majority of cases, the frameworks address all four RAI principles. Nevertheless, there are frameworks that cover only one (15.49%) or two principles (15.49%). In particular, when the principles addressed are three, the most covered are *Diversity & Non-discrimination & Fairness, Privacy & Data Governance, and Transparency* (9.3% for COMPANIES, 15.8% for UNIVERSITIES and 14.1% for NPG/COMM/PE). While, for two principles, the most covered are *Diversity & Non-discrimination & Fairness and Transparency* (9.3% for COMPANIES and 16.9% for NPG/COMM/PE). The interested reader can refer to the appendix for the complete results.

RQ3. *Do these frameworks provide recommendations for each phase of the Software Development Life Cycle (SDLC)?*

[15]Differently from the previous graph, here I counted all frameworks, even if they are proposed by the same entity.

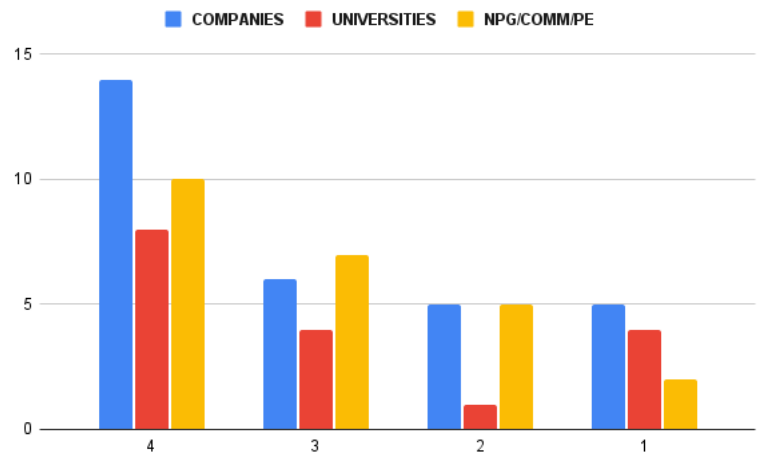


Figure 4.3: Number of RAI principles addressed by the frameworks grouped by proposing entity type.

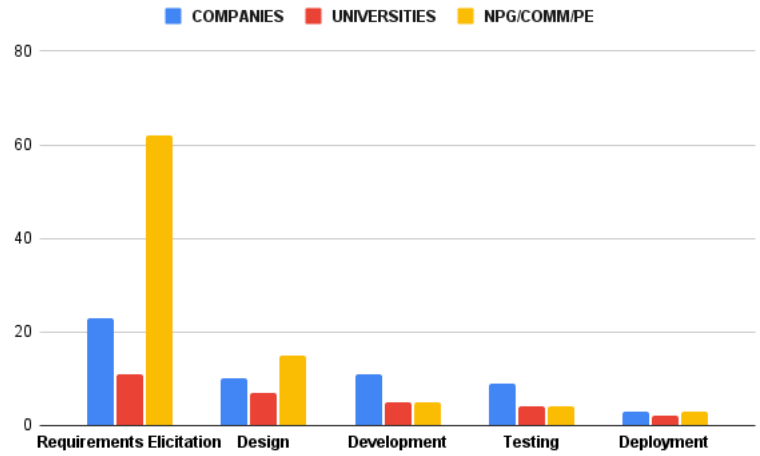


Figure 4.4: Distribution by SDLC phase addressed.

To calculate the statistics to answer RQ3, RAI principles have been mapped to project requirements (see Section 4.1.1) since they are actually high-level statements which put constraints on the project design. Moreover, I grouped resources proposed by each entity and mapped it w.r.t. the correspondent SDLC phase addressed. So, here, if an entity proposed two (or more) frameworks, it was counted once.

Fig. 4.4 shows that there is a common pattern regardless of the type of the proposing entity: more than half of the frameworks provide support only for the **Requirements Elicitation** phase (96/174, 55.17%), while subsequent phases are supported by very few frameworks. Anyway, COMPANIES showed more interest w.r.t. the other entities in also covering **Development** and **Testing** phases. Finally, the **Deployment** phase is the least supported by the frameworks, regardless of the type of proposing entity.

In Fig. 4.5 I investigate how much of the SDLC is covered (i.e. how many phases are addressed) by each framework. Here, I counted the frameworks, not only the proponent entities.

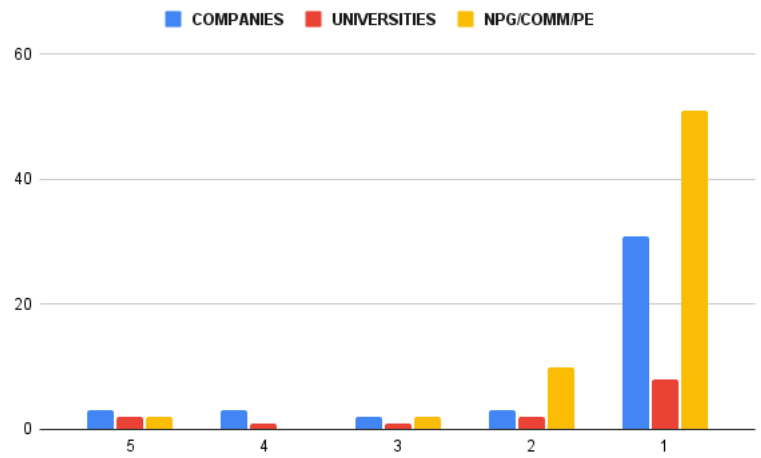


Figure 4.5: Number of SDLC phases addressed by the frameworks grouped by proposing entity type.

For frameworks proposed by COMPANIES sometimes I was not able to perform this mapping as no details on the internal core operations were provided. This is due to the fact that they offered the algorithm audit service as a lucrative service, so they are "closed-source" frameworks. In such cases, I excluded the frameworks from the statistics elaboration.

What emerges is clear: most frameworks provide suggestions for just one of the five SDLC phases: **Requirements Elicitation**, while the second most supported phase **Design**. Only 5.79% of the frameworks supports all the SDLC phases.

RQ4. *Is there a supporting tool for each proposed framework?*

I highlight that in this analysis the presence of a supporting tool can be labelled with four values: "Yes", "No", "Partially" and "Not Mentioned".

Partially is used for:

- Resources that cannot be properly considered tools; e.g. Excel sheets
- Tools that cover only a small part of the entire knowledge provided by the framework; e.g. a tool which covers only **Development** phase and neglects **Requirements** and **Design** phases

In the computation of values, I ignored the frameworks containing **Not mentioned** values for tools.

Regarding the statistics, the results are quite clear: as shown in Fig. 4.6 in more than the 80% of the cases there is no tool included in the framework. This is true regardless of the proposing entity as shown in Fig. 4.7.

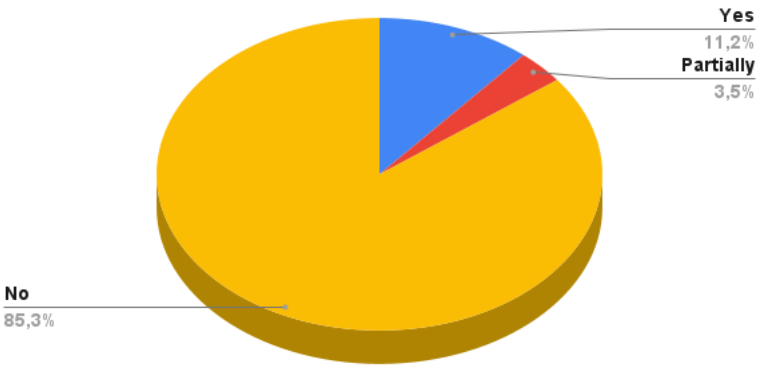


Figure 4.6: Distribution by the existence of a supporting tool regardless of the proposing entity.

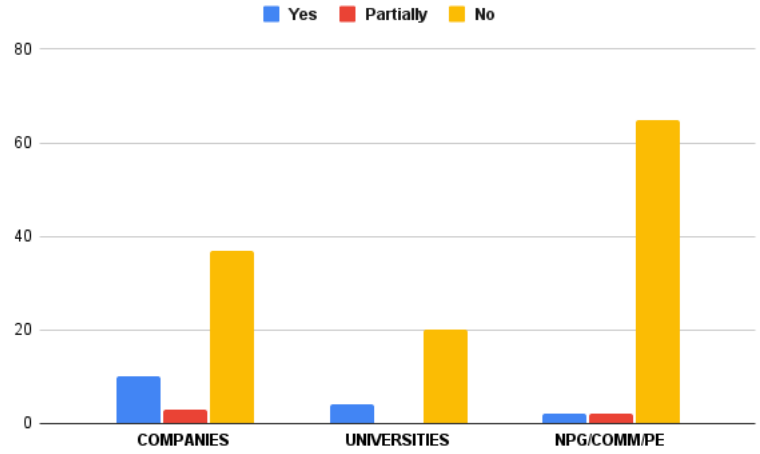


Figure 4.7: Distribution by the existence of a supporting tool and proposing entity.

Moreover, Fig. 4.8 shows, when a tool is present, by which type of entity it is provided. Here the interesting result is that more than half of the tools are proposed by COMPANIES (61.9%, 13/21).

Then, to go deeper into this investigation, I analyzed which kind of stakeholder's background is required to use these tools. I used the tag **Technical** to describe tools that should be used by stakeholders who work in the last three phases of the SDLC (e.g. developers, testers, IT technicians, etc); while I used the tag **Non-technical** for tools that should be used by stakeholders who work in the initial two phases of the SDLC (e.g. commercial agents, functional analysts, architecture designers, etc). Finally, if there are tools that offer resources for both kinds of stakeholders, I used the tag **Both**. Fig. 4.9 shows that, regardless of the type of proposing entity, more than half of the tools can be used by **Non-technical** stakeholders (57.4%, 27/47). Nevertheless, a significant amount of tools for supporting the **Technical** stakeholders is also provided (29.8%, 14/47). The tools that can support both kinds of stakeholders, conversely, represent only a very low percentage (12.8%, 6/47).

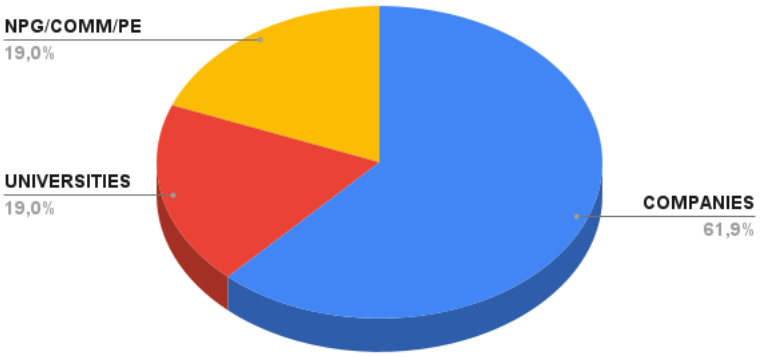


Figure 4.8: Distribution by proposing entity in case a tool is provided.

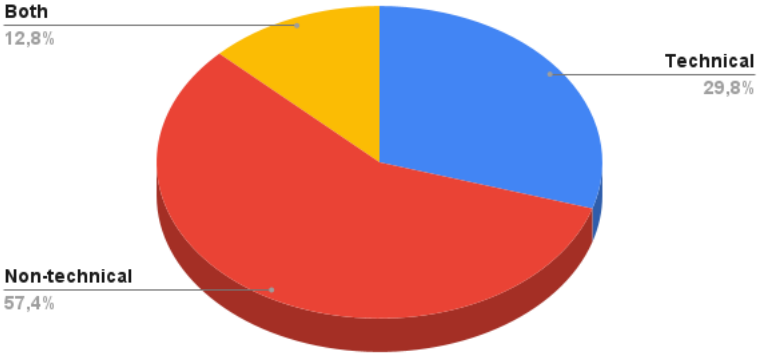


Figure 4.9: Distribution of stakeholder's required background regardless of proposing entity in case a tool is provided.

Furthermore, Fig. 4.10 shows that a significant amount of tools proposed by COMPANIES can be used by both types of stakeholders (33.33%) while UNIVERSITIES and NPG/COMM/PE proposed mainly tools that can be used by only one type of stakeholders.

Discussion

RQ1. What are the Responsible AI frameworks proposed in the literature? The RAI frameworks found in this search are proposed by different types of entities. The difference in the percentages shown in Fig. 4.1, depending on the type of entity, may be due to various reasons, among others: the different amount of funds available for research (and implementation) and the Return-on-investment (ROI) of these kinds of initiatives, e.g. different between companies and no-profit organizations. Moreover, it is noteworthy that the group of NPG/COMM/PE is the largest and most heterogeneous group among the three, including as well scientific works conducted as academy-industry collaborations.

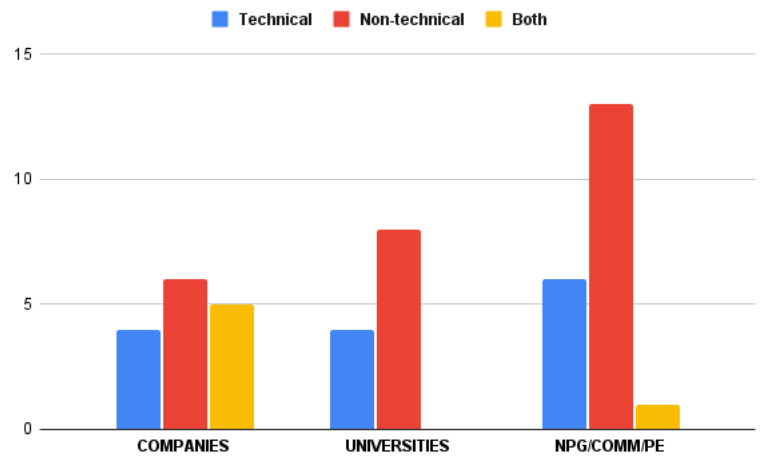


Figure 4.10: Distribution of stakeholder's required background by proposing entity in case a tool is provided.

Regarding the way in which the frameworks are categorized, we can definitely say that there is a worrying lack of tools: most of the frameworks provide just **Principles** or **Guidelines**. I want to point this out since it is difficult to translate principles into concrete and measurable actions [289] but a tool may help reduce this gap. Furthermore, without tools or well-known practices, it is not possible to measure the goodness and sufficiency of the adopted practices.

RQ2. How much do these frameworks address the various RAI principles? In addressing this research question, my aim was to discover how much "coverage" the frameworks give in terms of ethical principles handled. Indeed, sometimes practitioners are forced to use more than one framework at the same time in order to take into account the different dimensions of the RAI they want to implement and/or validate.

The emerging trend seems positive: the majority of the frameworks address all the four most important RAI principles, even if sometimes in a "partial" way (the principle is not fully-covered, see Section 4.1.1). Nevertheless, there are frameworks that neglect one, two, or even three principles. This could be due to multiple reasons, first of all, the fact that there is no global consensus about a clear definition of RAI and about which principles should be covered or implemented among the others. Moreover, especially for companies, there could be reasons leading to ignoring part of the principles: among these, the *Ethics Bluewashing* [290] (the unethical behavior of making unsupported or false claims about, or taking insufficient action to support, the ethical ideals and advantages of digital processes, products, services, or other solutions in an effort to look more ethically conscientious than one actually is) and the *Digital ethics shopping* (the unethical practice of selecting, modifying, or updating ethical guidelines, codes, frameworks, or other similar standards from a variety of offers in order to retrofit some pre-existing behaviors and thereby justify them a posteriori, rather than putting new behaviors into practice or improving them by comparing them to accepted ethical norms [290]).

RQ3. Do these frameworks provide recommendations for each phase of the Software Development Life Cycle (SDLC)? The RR highlighted that most frameworks

focus only on the initial phases of the SDLC, and in particular on *Requirements elicitation*. This is a problem because all the statements and obligations provided as RAI principles must result in operations and practices actually implemented while developing an AI application, encompassing the entire SDLC. Indeed, for developers, testers, and system installers is really difficult to identify practices that can satisfy these abstract requirements without any support from an expert in the field or from a practical framework guiding them in all the development phases. What sounds alarming is a common trend across all the different types of entities, as the highest percentages of proposed frameworks fall in the *Requirements Elicitation* phase while, from *Design* onward, the numbers drop dramatically until the lows in the *Deployment* phase.

RQ4. Is there a supporting tool for each proposed framework? Here there is an emerging negative trend: in most cases, there is not a practical tool complementing the theoretical framework proposed and this is valid regardless of the type of entity releasing the tool.

Moreover, most of the tools are Excel sheets, checklists, or questions to self-ask in the initial phases of the SDLC, so tools are mainly useful just for non-technical stakeholders. However, differently from the other entities, companies proposed a significant amount of tools useful for both kinds of stakeholders: technical and non-technical ones. This may be due to an interest in trying to sell complete services to other companies supporting every kind of stakeholder and, at the same time, covering every phase of the SDLC.

Summary of findings. A global perspective on the results analyzed leads us to affirm that there is not a "catching-all" framework, simple and complete enough to provide different levels of knowledge for different kinds of stakeholders (technical and non-technical ones), which can simplify and speed up the adoption of RAI practices. This statement comes from the fact that there are no frameworks encompassing all RAI concepts logically and uniformly: recommendations are scattered across various pages or documents, and there is no one element that acts as a glue and makes the information consultation linear and fluid, at various levels of abstraction, for various types of stakeholders. To better explain this concept, we can take the work of Baldassarre et al. [291] as an example: here, each element which composes the knowledge base is connected and this makes it possible to consult the knowledge base from various entry points, following different mental paths. As a consequence, it is advised the need for the creation of a uniform framework also for RAI: this way, the integration of RAI practices would be far easy in every kind of organization, public or private.

4.2 AI-enabled practitioners' needs and challenges

This section describes the work conducted in [7].
This work builds on the results of the previous review and aims to investigate how AI professionals deal with TAI issues on a day-to-day basis. To better understand practitioners' needs I distributed a **survey** and conducted semi-structured one-on-one **interviews**, collecting data from a total

Table 4.4: Activities integrated into the traditional SDLC phases to support AI-enabled systems development.

Traditional SDLC phase	**Integrated activity**
Requirements Elicitation	Model Requirement
Design	Data Collection, Data Preparation
Development	Feature Engineering, Model Training
Test	Model Evaluation
Deployment	Model Deployment
Monitoring	Model Monitoring

of **34 practitioners** employed in companies of different sizes.

As the main contribution, this work intends to deeply investigate practitioners' views, needs, and challenges in developing TAI systems throughout the entire SDLC. The novel contribution can be summarized as follows:

- I have analyzed the **existing procedures** that development teams adopt when implementing Trustworthiness in AI;
- I have investigated the **impediments** encountered in the attempt to implement Trustworthiness in AI;
- I have identified a range of practitioners' **needs** that academic and industrial research should seek to address.

This study has identified a range of practitioner needs that have thus far been overlooked in the literature. For example, the majority of the interviewees report a lack of tools for the late stages of the SDLC, as well as of knowledge bases and practical guidelines with suggestions on implementing TAI across the entire SDLC.

Method

The goal of this study is to investigate the **state of the practice** to understand common practices as well as challenges and difficulties encountered by AI practitioners in implementing TAI systems through the entire SDLC.

Since traditional phases of the SDLC do not necessarily map well with the activities required to develop an AI-enabled system, I have extended each phase with one or more activities mentioned by Zhengxin et al. [292]. Table 4.4 graphically shows how each traditional SDLC phase's definition has been extended.

Given this main goal, and based on the analysis of the shortcomings and common practices that emerged from the mapping study carried out in my previous work [4], I defined the following Research Questions (RQs):

- **RQ1**: What vision/opinion do practitioners have about untrustworthiness issues and how often do they encounter them?
- **RQ2**: How do practitioners address untrustworthiness issues?
- **RQ3**: What tools/support do practitioners desire to address untrustworthiness problems better?

Data collection

For this study, I recruited practitioners[16] working on AI products and services through a combination of purposive and snowball sampling [293]. To recruit participants for this study, I chose to not widely distribute the survey online, but rather to conduct a brief screening procedure to define the inclusion criteria and target suitable participants for the study. The target participants for the survey were AI practitioners involved in the development of AI-based systems, with at least some basic knowledge of TAI principles and/or who had previously addressed TAI in their professional work. I started by sending personal emails to contacts within my network, working in industry or academia, explaining the purpose of the study and the inclusion criteria; then, I asked them to help me recruit other participants by spreading the invitation through their networks. Next, I verified that the inclusion criteria were met by asking some specific questions in the demographics section of the survey.

The invitation to participate in the study was sent by email and included an explanation of the study's purpose. In order to meet all needs (*e.g.*, restrictions related to tight schedules, time zones, commitments), I gave participants the chance to choose between (a) survey (asynchronous interaction) or (b) semi-structured interviews (synchronous interaction) mode. This allowed me to include a broader amount of subjects and collect a higher number of answers. All participants answered the same set of questions and had the opportunity to add any non-listed practices or suggestions/feedback related to the closed-ended question in the open-text fields (asynchronous mode) whereas, interviewed participants (synchronous) could "discuss their answers out loud" and further elaborate their considerations with the interviewers. The answers were all transcribed to be included later in the thematic analysis.

Participation was on a voluntary basis and not rewarded by any means. The survey can be accessed at [294]. Overall, I obtained **23 answers** for the **survey** and **11 participants** attended the **interview**.

[16]By "AI practitioners" I mean those who work in any role on a team developing products or services involving AI.

Section 4.2 provides details about participants' demographics and their relevant experience. Specific details about their companies and working environment have been abstracted to preserve anonymity.

The survey

The survey contains six main sections [294]:

1. *Informed consent request.* This page asks the participants to provide their informed consent and explains the purpose of the research, the participants' requirements, confidentiality rules, participation on a voluntary basis, and the time needed to complete the survey.

2. *Preliminary concepts knowledge.* I clarified the semantics and interpretation of each TAI principle for participants by providing a definition for each principle. By listing a definition, I wanted to build a shared understanding of each principle to answer the remaining questions. In addition, I asked the participants about their vision and previous experience with TAI.

3. *Practices in preventing untrustworthiness in AI.* I inquired participants about the main strategies they adopt to prevent TAI issues (*e.g.*, balancing the dataset or choosing a specific algorithm).

4. *Practices in discovering untrustworthiness issues in AI.* I asked participants to share their strategies to find possible sources of untrustworthiness (*e.g.*, do auditing tasks, compute metrics, learn from user feedback).

5. *Practices in addressing trustworthiness issues.* I investigated the different approaches used to address TAI issues (*e.g.*, dataset augmentation, instance weighting).

6. *Demographics and background information.* Participants were asked about gender, level of education, country, role, years of experience, size of the organization, etc.

All the questions of the survey sections (3), (4), (5), illustrated above, as well as the options for the answers, were inspired from and based on my previous work [4], which set current shortcomings and common practices in literature.

The survey was anonymous and did not ask for any directly identifying information. Most of the survey questions were closed-answer and mandatory, but there were also optional open-text ones. Through the latter, I was able to collect further qualitative data as many of the participants provided information on the practices they usually implement. All survey data and raw material can be accessed in the online appendix [295].

Interview Study Protocol

Before scheduling the interviews with participants, to understand the challenges and requirements for conducting remotely semi-structured interviews, as well as to help us refine the study protocol, I conducted two pilot interviews. These preliminary interviews helped me define the protocol and the setting I applied to the final, larger sample.

All interviews were conducted via Microsoft Teams[17]; after asking each participant for their consent, I enabled *Recording & Transcription* Teams features. Once the interview ended, I proof-checked the transcription in order to correct any misspellings, anonymized any Personal Identifiable Information (PII) and finally deleted the recording.

The interview study consisted of think-aloud semi-structured interviews, each one lasting between 45 and 90 minutes. During the live interview, I periodically asked participants to elaborate on their responses, especially for the open ones. I also encouraged participants to "think aloud" [296, 297] and discuss the information that was being displayed and how their understanding of the question was developing.

To give a standard structure to each interview, I used the survey as a canvas.

Data Analysis

In this step, I extracted all relevant data using quantitative and qualitative data analysis techniques to summarize and interpret the collected data. For quantitative data, I used descriptive statistics [298], and for qualitative data, I used thematic analysis [299].

I used an **inductive thematic analysis** approach [300, 301] to analyze about 11.5 hours of video recordings and their corresponding (automatically generated and manually proof-checked) transcripts. The entire analysis was done through Atlas.ti[18]. I and another researcher worked independently and used the tool to conduct an open coding of the transcripts for each quotation. Next, we manually reviewed each code and decided which to include/exclude annotating any comments. Once this step was completed, we joined to compare and discuss results. The total number of analyzed quotations was 23. The calculated Cohen's Kappa [302] is 0.259. All the details about the coding procedure and the generated codes are provided in the online appendix [295].

Once finalized, the codes were shared with the entire research team and grouped into higher-level themes concerning the practitioners' knowledge and practices. In next Section I discuss the findings identified from these codes and themes, together with implications for future TAI developments.

Results and findings

I present findings from the think-aloud interviews study and the survey answers, divided into three main sections,

- Practices in **preventing** untrustworthiness in AI
- Practices in **discovering** untrustworthiness issues
- Practices in **addressing** untrustworthiness issues

[17] `https://www.microsoft.com/it-it/microsoft-teams`
[18] `https://atlasti.com/`

Table 4.5: Reasons why participants care about TAI principles.

Reason	Disagreement (1-2)	Neutral (3)	Agreement (4-5)	N/A
Doing something about trustworthiness in AI/ML	4	6	23	1
Avoid violating legal requirements	2	3	27	2
Avoid reputational damages	4	8	21	1
Improve the overall quality	3	3	26	2
Retain users/avoid losing business	8	4	19	3

Across all three phases, I discovered different nuances of practitioners' needs around TAI issues. I supplement data from the closed-ended questions (quantitative results) with the thematic analysis performed on the answers from all the open-ended questions (qualitative results). I performed analysis on the disaggregated data with respect to subgroups such as company size, gender, education, and number of projects deployed. To understand if the differences were statistically significant, I conducted pairwise comparisons using Fisher's exact test coupled with the *Benjamini-Hochberg* [303] correction to obtain the adjusted p-values. Since there is no statistical significance in any of the cases except for company size, detailed in Section 4.2, the following graphs report the results of the analyses in aggregated form.

Preliminary concepts knowledge

All discussed tables and graphs, from now on, bring together the answers from both the interviews (11) and the survey (23). In the survey section "2. *Preliminary concepts knowledge*", I observed that the TAI principle participants have encountered most frequently in their projects is **Privacy** (20 answers), followed by **Transparency** (18 answers) and **Security** (17 answers), while the least experienced is **Fairness** (13 answers). This answer should be further investigated because perhaps sometimes practitioners may not recognize or be aware of the need to address some issues related to these principles.

In addition, Table 4.5 shows the reasons participants agreed on concerning why they care about TAI. The most agreed reasons were *Avoid violating legal requirements* and *Improve the overall quality*. While, the least agreed one was *Retain users/avoid losing the activity*, with eight disagreements.

Other important factors related to the reasons for caring about TAI emerged: i) "*need to solve mission-critical tasks*"; ii) "*[need to provide] models usable in real-world contexts*", and this demonstrated that black box models are not allowed in some specific contexts; iii) "*the robustness of the AI explanations themselves*", which shows consciousness about the fact that all TAI aspects contribute to making the model more robust; and iv) "*[need to] desire to commercially*

assemble AI systems to improve society".

Finally, Fig. 4.11 shows that most of the participants address TAI principles during the **Design** and **Development** SDLC phases. In contrast, very few participants reported that they had addressed TAI principles during the **Requirements Elicitation** and, especially, **Deploy** phases.

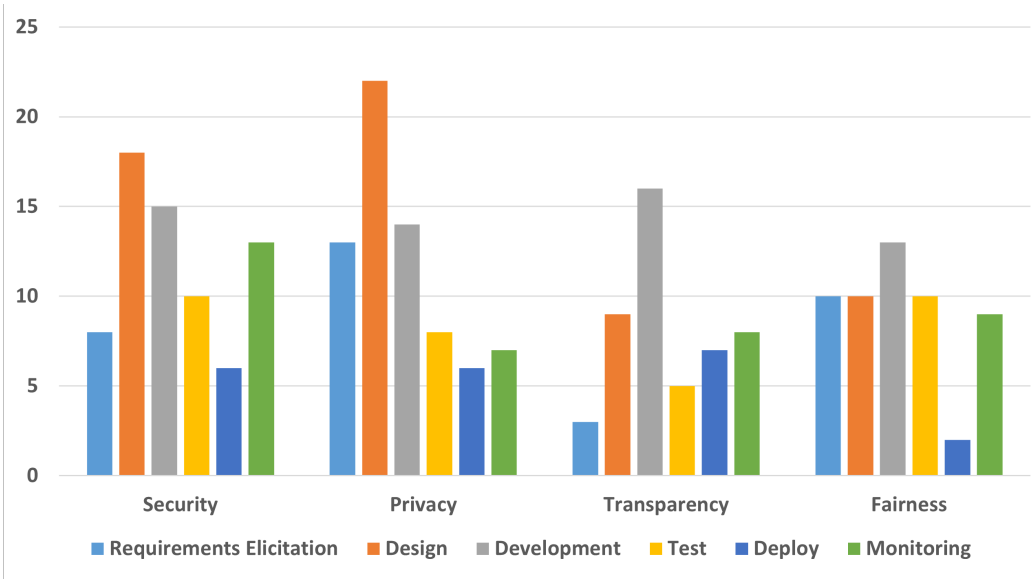


Figure 4.11: TAI principles addressed by SDLC phase.

Practices in Preventing untrustworthiness in AI

In the survey section "3.*Practices in preventing untrustworthiness in AI*", as Fig. 4.12 shows, the most recurrent strategy employed by the participants to ensure trustworthiness is "*algorithm that can best explain the decision*". On the other hand, the least employed practice appears to be "*inject malicious data points*". In analyzing the disaggregated data, I found a statistically significant difference only in the responses related to the strategy "*Algorithm that can best explain the decision*" for the company size subgroup. Indeed, for medium-sized companies, I found more positive responses than for small and large enterprises. For medium ones, no negative answers were given and 63% of the participants chose "Always", reflecting the wide use of this strategy in medium-sized companies.

Some participants mentioned other strategies, such as "*[conduct an] in-depth study of the state of the art [prior to start designing the system]*" and "*[use the] post-processing phase [...] to apply human-friendly deterministic rules to check whether a result is in line with the sense of the application domain*".

Additionally, when I asked the participants to rate the utility of various hypothetical tools assuming their team had access to them, the participants rated as the most valuable the tool able to "*[...] generate an explanation of a model after its creation [...]*". On the other hand, they rated as least useful the tool to "*decide how much data you need for particular subgroups/subpopulations*". These results are shown in Fig. 4.13.

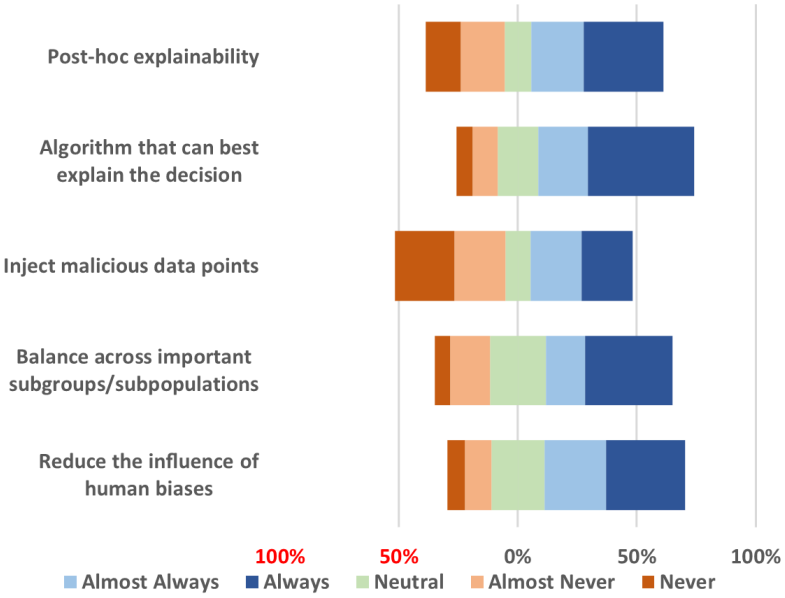


Figure 4.12: Strategies employed to ensure trustworthiness in AI. N/A answers have been removed.

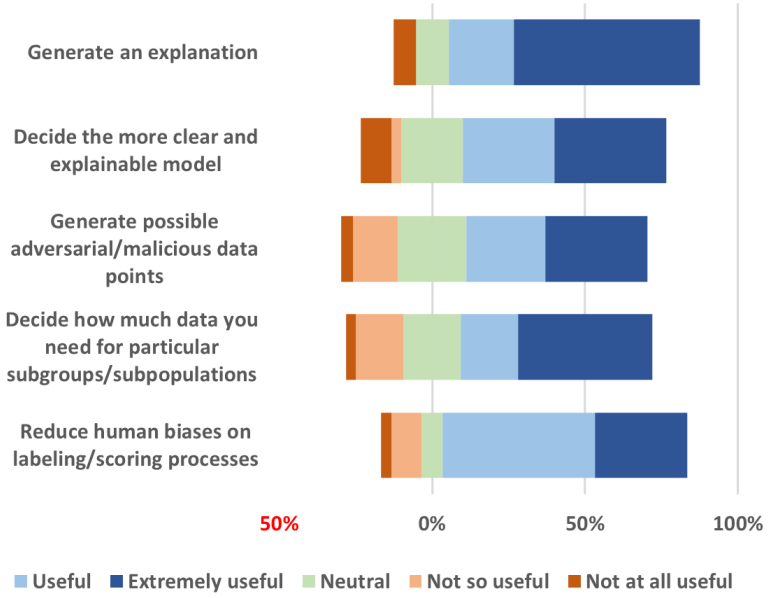


Figure 4.13: Perceived usefulness of hypothetical tools to prevent TAI issues. N/A answers have been removed.

Here too, some important insights emerged, including "*tools to improve software architectures*" and "*[a tool for] referencing the best architecture (dependencies, docker files, instant compute, ...) to perform the task with the lowest possible costs*".

The answers to this section show practitioners are prone to use techniques and tools to prevent trustworthiness issues, focusing mainly on ensuring **Transparency** (a.k.a. Explainability).

Practices in Discovering Untrustworthiness Issues

In the survey section "4.*Practices in Discovering untrustworthiness in AI*", I investigated which strategies participants mainly employed to discover TAI issues.

The data shows that the most used strategies are "*Metrics/KPIs*", "*learn from user feedback*", and "*examine AI/ML model's input features*" (see Fig.4.14). Examples of *Metrics/KPIs* related to fairness are, just to cite a few, Demographic Parity, Accuracy, F1-Score. Whereas, *user feedback* is intended as having a feedback form where the users can report misbehavior by the algorithm [304]. What stands out is that the strategy less employed by the participants is "*generate specific adversarial/malicious samples*" (4/10 negative answers).

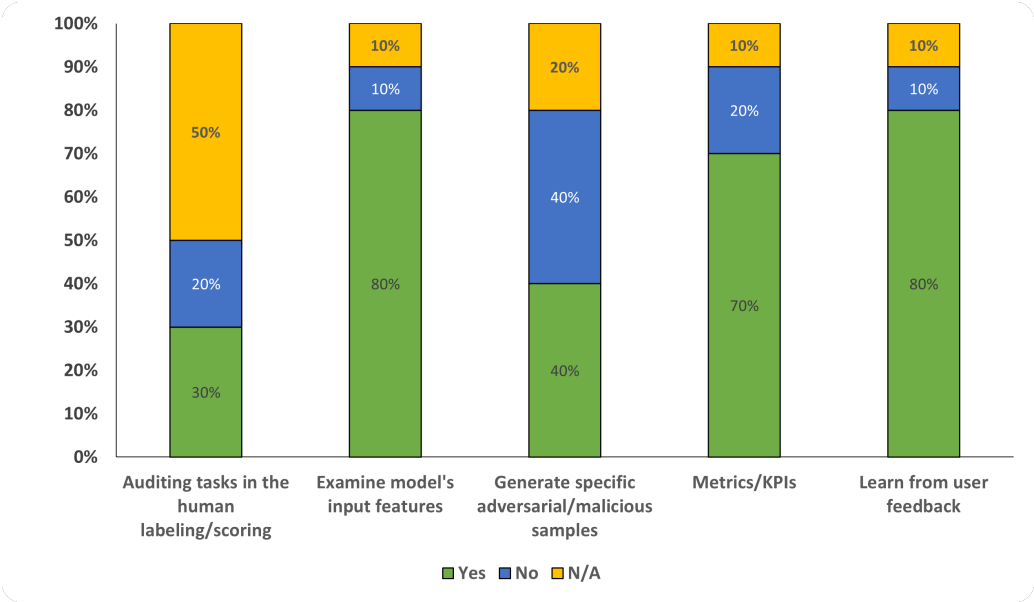


Figure 4.14: Strategies employed to discover untrustworthiness in AI.

Moreover, when asked for other (not mentioned) strategies they employ, one participant answered "*post processing studies to evaluate possible model 'discriminations'*". As a free thought, another declared "*apply confidence criteria such that it is possible to measure how often the model fails to respond reliably*".

What emerges in this section is that participants employ both qualitative (*e.g., auditing tasks in the human labeling/scoring process*) and quantitative (*e.g., metrics and KPIs*) strategies.

Practices in Addressing Untrustworthiness Issues

Regarding survey section "5.*Practices in Addressing Untrustworthiness in AI*", participants reported that after finding a TAI issue only in 35% of the cases (12/34) the team addressed it directly, while 15% (5/34) of the participants stated that it was not addressed by them, but handled by a third party. Worth noting is the fact that in 50% (17/34) of the cases participants reported that they did not fix the issue after finding it. The reasons why participants did not solve the issue after finding it are asked in a subsequent question (see Table 4.6).

Table 4.6: Reasons which prevented participants from addressing/fixing AI trustworthiness issues.

Impediment	**Yes**	**No**	**N/A**
No one had idea on how to solve the issue	0	9	3
The issue solution required high human effort, which we could not afford	5	5	2
The issue solution was too expensive (financially) to address	4	5	3
The issue solution required too much time to be implemented	7	3	2
The issue solution was likely to decrease the performance of the system (*e.g.*, decreasing accuracy)	6	3	3
There was not a tool which automated the fixing process	5	5	2

When participants addressed any issues found, they declared the most implemented strategies were "*improving the quality of the dataset (e.g., removing spurious samples, paradoxical values)*" (8 answers) and "*augmenting the dataset (e.g., with artificial, manually generated data points)*" (6 answers). On the other hand, the less implemented strategy was "*searching for a tool which automates a specific trustworthiness issue-fixing process*" (1 answer).

One interviewee also mentioned that they usually approach explainability by "*using [only] white box models*". More details can be found in Figure A2 in the online appendix [295].

Regarding the reasons why participants did not solve a TAI issue after finding it, Table 4.6 shows the most frequent reasons are "*the issue solution required too much time to be implemented*" (58.3%) and "*the issue solution was likely to decrease the performance of the system (e.g., decreasing accuracy)*" (50%). On the other hand, none of the participants answered: "*no one had an idea on how to solve the issue*"; this is a positive result since demonstrates practitioners are conscious of untrustworthiness problems and can formulate hypotheses on how to address them. During the interviews, one participant also mentioned "*data availability*" as an impediment.

Finally, when I again asked the participants to rate the utility of various hypothetical tools — assuming their team had access to them — the participants rated as the most valuable a tool able to "*[...] help [...] monitoring the AI model after its release to the public*", followed by "*best practices that can actively guide your team through the model's SDLC*", "*tools to help the team in the data pre-processing steps (e.g., decide whether one needs to add/remove data points from your training set, and what kind of data you need to add/remove)*", and "*a knowledge book in which are mapped trustworthiness problems and [...] solutions*". On the other hand, they rated as least useful tools "*[...] to help your team doing an ex-post TAI audit*" and tools able to *[...] help your team deciding which AI model best respects the TAI principles [...].* These results are graphically shown in Fig. 4.15.

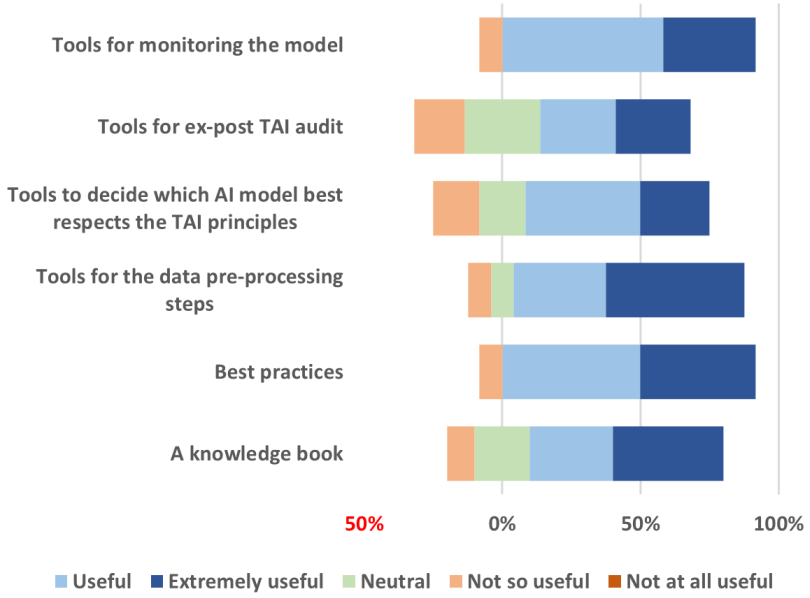


Figure 4.15: Perceived usefulness of hypothetical tools to address TAI issues. N/A answers have been removed.

Demographics and background information

In terms of demographics of the sample, the study participants' gender is represented by 68% male, 21% female, 3% non-binary or gender diverse, and 9% preferred not to respond. Concerning academic qualifications, 56% of the participants have a master's degree, 41% have earned a Ph.D., and 3% have completed their bachelor's studies in computer science-related fields. The notable prevalence of advanced education within the respondent pool aligns with the expectations, reflecting the elevated cognitive expertise required by the complexities of this particular field.

The vast majority of the participants are employed in medium-large companies, specifically, 32.3% (11/34) work for companies with less than 50 employees (small), 23.5%, (8/34) for companies between 50 and 500 employees (medium), while 44.1%, (15/34) are employed in large companies with more than 500 employees.

Participants have on average five years of experience in their role and two years of experience in the AI field.

Regarding the technology area that best describes with which AI products/services the participants work, the four most prevalent are "*Decision support*" (15 answers), "*Natural Language Processing*" (13 answers), "*Computer Vision / Image Analysis*" (12 answers), and "*Recommender Systems*" (9 answers).

Finally, regarding the number of AI-enabled projects developed and deployed into a production environment, I observed that most of the participants (19/34, 56%) declared that just a small percentage of the developed projects — from 1 to 30% — are deployed in a production environment, while only 3% of the participants declared that most of the projects — from 90 to

100% — are deployed into a production environment. This reveals the fact that most of these types of projects are still in an experimental stage.

Due to space constraints, I have not included tabular representation of demographics in the paper which are, however, all available in the online appendix [295][19].

Summary of key findings

Here I summarize some key findings from this study.

F1. The study reveals that participants care a lot about **Privacy** and **Transparency**. Indeed, among the most used strategies to ensure trustworthiness are "*post-hoc explainability*" and "*algorithm that can best explain the decision*" (Fig. 4.12). In addition, tools that "*generate an explanation*" and that help in deciding "*the more clear and explainable model*" are among the tools perceived as most useful (Fig. 4.13).

F2. Acting on the **dataset** is one of the most used strategies to solve the found TAI issues. Indeed the most implemented strategies are "*improving the quality of the dataset (e.g., removing spurious samples, paradoxical values)*" and "*augmenting the dataset (e.g., with artificial, manually generated data points)*".

F3. Business constraints — like the time required to implement the solution or the unacceptable performance drop — often represent **impediments** to implementing trustworthy AI applications (see Table 4.6).

F4. Analyzing the **tools** practitioners lack the most in addressing TAI issues, a need for tools for after-deploy **monitoring** and **best practices** and TAI **knowledge books** that can actively guide a team through the SDLC emerges (Fig. 4.15).

F5. Many TAI projects are developed but not deployed in a production environment, which reveals that in some cases practitioners are still experimenting with this field.

Discussion

RQ1. What vision/opinion do practitioners have about untrustworthiness problems and how often do they address them?

The findings reveal that the most addressed principle is **Privacy**, probably because it is contained in various regulations that exist and must necessarily be complied with (*e.g.*, in Europe the GDPR [305]). **Transparency** is also often taken into consideration, probably because there are domains where it is a fundamental and unavoidable feature required by the law, such as in Healthcare and Financial Services. On the other hand, the one less addressed is **Fairness**, perhaps because there are still no clear and shared regulations for this dimension of trustworthiness and everything is left to the initiative and ethical values of those implementing these systems. As a result, even large and well-established companies in the industry are often

[19]Appendix Table A4 (interviewees' self-reported technology areas and team roles); Table A5 (participants count grouped by company size); Table A6 (participants count grouped by years of experience in their current role and in developing AI-enabled systems)

caught up in scandals that damage their reputation and show how even the most popular and widely used algorithms suffer from unfairness[20].

It is notable that when I asked practitioners *why their team cares about trustworthiness in AI*, the motivation "*retain users/avoid losing business*" found most disagreement among them. This may reveal that they believe that TAI issues do not lead to losing users and/or business. Moreover, during interviews, it emerged that addressing TAI is, in some cases, even mandatory and not an option.

Finally, this study suggests that TAI is mainly addressed in the early stages of the SDLC. While this is good — since the earlier certain decisions are made, the more effective they are in the design of the final model — this also reveals that practitioners are most likely not aware of tools and best practices to be used in the final stages of the lifecycle. In fact, **Deploy** is one of the least addressed phases. Indeed, one interviewee also mentioned the need for guidelines on the best cloud provider compliant with TAI practices. These elements allow us to infer that the choice of deployment infrastructure is often left to chance or, in the best case, to routines and/or trust in a specific cloud provider.

RQ2. What do practitioners do to address untrustworthiness problems?

Based on the answers, it is clear that when practitioners want to solve TAI issues they mainly act on the dataset, as the most implemented strategies are related to "improving the quality of the dataset" and "augmenting it with artificial data points". As shown by Fig. 4.11 and Fig. 4.14, TAI is mainly addressed in **Design** (*e.g.*, examining model's input features) and **Development** phase, without disregarding the **Monitoring** one (*e.g.*, learning from user feedback). Answers to question 22[21] sustain this trend and highlight that practitioners also give little consideration to the possibility of searching for tools that automate their manual activities, perhaps because they feel safer with manual analyses or because they are not familiar with automatic tools. Finally, answers to question 24[22] demonstrate that business constraints (*e.g.*, time, money) often hinder the resolution of TAI issues: if solving a problem takes too long or may cause a slight performance degradation, practitioners tend to not address it.

RQ3. What do practitioners desire to better address untrustworthiness problems?

What emerged from the data is that the tools most in demand are those that can generate an explanation of the model after its training. However, literature provides several mature models to explain both traditional ML algorithms — see SHAP [306] and LIME [307] — and neural networks — such as [308]. Perhaps these answers could be due either to the unsuitability of such tools for specific issues to be solved or to scarce knowledge about explainability tools landscape, which would also explain the fact that *post-hoc explainability* strategies are infrequently used (see Fig. 4.12). On the contrary, the tools deemed less useful are the ones that help decide

[20] `https://www.bloomberg.com/graphics/2023-generative-ai-bias/`

[21] Q22: Which of the following strategies has your team evaluated, and which strategies were actually implemented?

[22] Q24: What prevented your team from addressing/fixing these AI trustworthiness issues?

how much data they need for particular subgroups/subpopulations, probably because there are other factors — not related to TAI requirements — that constrain practitioners while collecting and/or pre-processing the data. For instance, sometimes, collecting more data about a specific sub-group could be simply infeasible or very difficult.

Furthermore, it is evident that practitioners feel the need for tools to monitor the model after it has been released to the public, they especially express a need for guidelines and a knowledge base to help them in implementing TAI throughout the SDLC. Moreover, practitioners believe a single tool *to perform post-hoc analyses* to be of little use, probably because they feel that it is too late to worry about TAI once the system has been released on the market to end users.

Practical implications and recommendations

Based on the previous discussion, I summarize some key practical implications and recommendations for the AI industry and the AI research community.

P1. Practitioners should take TAI principles into account throughout the entire SDLC and not just in the early stages, such as the **Design** phase, as shown by Fig. 4.11 and answers to question 18[23]. Even if valuable mitigation measures can be put in place early in SDLC, there is a huge amount of mitigations that can only be implemented in the latter stages of the SDLC. For instance, using a tool that helps to choose the best cloud provider to be TAI-compliant.

P2. This study reveals that, on numerous occasions, even when TAI issues are identified, they are not addressed by practitioners — either directly or by third parties — due to business constraints such as limited time, financial constraints, or declining performance. Nevertheless, the oversight of these issues may result in significant economic and reputational consequences, incurring substantial costs for companies. Hence, it is imperative for the industry to commit to addressing TAI issues detected at various stages of the SDLC. In addition, such actions are crucial for compliance with emerging regulations, such as the AI Act [309].

P3. As a general remark, there is a pressing need for **guidelines**, **knowledge bases**, and **tools** that can help practitioners implement TAI principles throughout the entire SDLC. They need guidance and practical advice on which tools to use at each stage of SDLC and to address which principles. Often, as pointed out in my study, although these tools exist, practitioners may not be aware of them. Moreover, as pointed out in my previous work [4], there is a significant gap that should be filled between high-level AI ethics principles and low-level concrete practices for practitioners. For this reason, as a research community, we should rethink how to design these guidelines and best practices, so that they are readily available and usable by professionals and provide actionable guidelines that can be put into practice while implementing trustworthy AI applications.

[23]Q18: Please indicate if you use one of the following strategies to discover trustworthiness issues.

4.3 POLARIS Framework

In response to the challenges and issues highlighted by the previous research's results, with the intent to fill the gap between theory and practice and to address stakeholder needs and shortcomings, I have developed a framework: POLARIS [6].

Indeed, POLARIS has been designed to provide actionable guidelines and tools in order to support stakeholders in addressing TAI principles throughout the entire Software Development Life Cycle (SDLC). POLARIS provides a significant amount of information, organized and linked into a comprehensive knowledge base that is designed to be expandable, with the possibility to easily add new knowledge.

In the next sections, I explain how I built the POLARIS knowledge base and how to navigate it.

4.3.1 Defining POLARIS Knowledge Base

In this section, I describe how I assembled the POLARIS Knowledge Base and the selection process used to choose the different knowledge sources representing the foundation of POLARIS.

I started from the frameworks analyzed in [4], I complemented the analysis with the results obtained from the survey, and then I identified among the existing knowledge sources (i.e. frameworks) those that met both of the following criteria:

1. Have actionable guidelines (and not only a simple high-level principles list)

2. Address all SDLC phases.

Regarding the last criterion, since SDLC phases do not always map with the activities required to develop an AI-enabled system, I have integrated each SDLC phase with AI-enabled activities established by Zhengxin et al. [292]. Table 4.4 shows how each SDLC phase has been integrated with AI-enabled system development activities.

After this first selection phase, I identified only three knowledge sources that meet both criteria (1) and (2). Then, I mapped each identified knowledge source to the corresponding TAI principle. For **Explainability** I selected Jin et al. - EUCA: the Explainable AI Framework [310], for **Fairness** I chose Amsterdam Intelligence - The Fairness Handbook [311]. For both **Privacy** and **Security**, I selected ENISA - Securing Machine Learning Algorithms [312].

Then, I refined this first selection by adding more knowledge sources that could complement the information provided by the primary ones initially selected. We started by selecting the frameworks that met *at least one* of the following criteria:

1. Have actionable guidelines (and not only a simple high-level principles list)

2. Address all SDLC phases.

Table 4.7: The identified *best knowledge source* for each TAI principle.

TAI Principle	Knowledge Source
Explainability	Jin et al. - EUCA: the Explainable AI Framework [310] Tensorflow - Responsible AI in your ML workflow [313] CSIRO - Responsible AI Pattern Catalogue [314]
Fairness	Amsterdam Intelligence - The Fairness Handbook [311]
Security	ENISA - Securing Machine Learning Algorithms [312] ICO - Guidance on AI and data protection [315] Tensorflow - Responsible AI in your ML workflow [313] Microsoft - Threat Modeling AI/ML Systems and Dependencies [316] CSIRO - Responsible AI Pattern Catalogue [314]
Privacy	ENISA - Securing Machine Learning Algorithms [312] ICO - Guidance on AI and data protection [315] Tensorflow - Responsible AI in your ML workflow [313] Microsoft - Threat Modeling AI/ML Systems and Dependencies [316] CSIRO - Responsible AI Pattern Catalogue [314]

We retrieved 10 additional knowledge sources that met at least one of the previous criteria. The table with all the 10 knowledge sources identified can be found in the online appendix [10]. Then, I performed a comparative analysis between each primary knowledge source already selected ([310], [311], [312]) and the new ones retrieved in this second iteration.

The results of the comparative analysis brought us to select four additional knowledge sources that could complement and expand the information provided by the first ones selected.

The additional frameworks selected were ICO's "*Guidance on AI and data protection*" [315], Tensorflow's "*Responsible AI in your ML workflow*" [313], the guidelines in Microsoft's "*Threat Modeling AI/ML Systems and Dependencies*" [316] and CSIRO's "*Responsible AI Pattern Catalogue*" [314]. Therefore, I used these additional knowledge sources to further extend the information provided by the primary ones. We ended up selecting **7 knowledge sources**. Table 4.7 shows the mapping between each TAI principle and the corresponding knowledge sources covering that principle.

4.3.2 Navigating POLARIS Knowledge Base

Having defined the POLARIS knowledge base, in this section I focus on how to navigate it. The goal of the proposed framework is to support stakeholders throughout the SDLC by suggesting concrete implementation strategies able to support and guide them in the development of TAI applications.

When applying the framework, the users will ultimately receive an *Action* to implement, that is, an actionable guideline that a stakeholder should consider and, if possible, implement while developing the AI-enabled software system to ensure compliance with the four TAI principles. The user can also choose to filter and apply only a subset of the suggested guidelines.

As of now, the first version of POLARIS has been structured as a filterable Excel array of sheets. There are **four** main **knowledge components**, one per each principle: (i) *Privacy*; (ii)

Security; (iii) *Fairness* and (iv) *Explainability*.

In proposing the structure of each Excel sheet, I was inspired by the ENISA framework [312] and then customized it according to my needs.

The two sheets that contain the knowledge for "**Privacy**" and "**Security**" are composed of the following six columns (Fig. 4.16 shows an excerpt of the security component).

Figure 4.16: Excerpt of the Security component navigation of the POLARIS framework. The whole content can be found in the online appendix [10].

SDLC Phase	Threat	Sub-Threat	Description	Vulnerability (consequence)	Action
Design	Poisoning	Label modification	An attack in which the attacker corrupts the labels of training data. This sub-threat is specific to Supervised Learning.	Use of unreliable sources to label data	**(Technical) Ensure reliable sources are used**: ML is a field in which the use of open-source elements is widespread (e.g., data for training, including labeled ones, models). The trust level of the different sources used should be assessed to prevent using compromise ones. For example: the project wants to use labeled images from a public library. Are the contributors sufficiently trusted to have confidence in the contained images or the quality of their labelling?
Development	Failure or malfunction of ML application	Denial of service due to inconsistent data or a sponge example	ML algorithms usually consider input data in a defined format to make their predictions. Thus, a denial of service could be caused by input data whose format is inappropriate. It may also happen that a malicious user of the model constructs an input data (a sponge example) specifically designed to increase the computation time of the model and thus potentially cause a denial of service.	Use of uncontrolled data	**(Techincal) Control all data used by the ML model** Data must be checked to ensure they will suit the model and limit the ingestion of malicious data: - Evaluate the trust level of the sources to check it's appropriate in the context of the application - Protect their integrity along the whole data supply chain - Their format and consistence are verified - Their content is checked for anomalies, automatically or manually (e.g. selective human control) - In the case of labeled data, the issuer of the label is trusted.

(1) SDLC Phase. The SDLC phase that the *Action* column applies to.

(2) Threat. Contains the list of threats, i.e. possible attacks that can be conducted against an AI-enabled system. Examples are *Evasion* and *Poisoning* attacks.

(3) Sub-Threat. In some specific cases, a threat can have a specific declination in a sub-characteristic. For example, the *Poisoning* attack can be declined in *Targeted Data Poisoning* and *Indiscriminate Data Poisoning*.

(4) Description. A textual description of the (Sub)Threat, which helps the stakeholder obtain coarse-grained details about the threat and understand the attacker's objective.

(5) Vulnerability (consequence). This is the immediate consequence of having a model vulnerable to a specific threat.

(6) Action. The corresponding action, or decision, that should be adopted to address a specific threat, based on the SDLC phase and threat selected, keeping in mind the vulnerability.

For example, a developer in the *Design* SDLC phase who is trying to address the vulnerabilities associated with *Poisoning* threat, may consult POLARIS and access the *Security* Excel sheet, select the *Poisoning* threat — and corresponding sub-threat, i.e. *Label modification* —, and obtain a description of the vulnerability associated to the (sub)threat and the action to take in order to mitigate the vulnerability, i.e. *ensure that reliable sources are used* (Fig. 4.16).

The sheet that contains the knowledge for "**Fairness**" is composed of 5 columns, all of the above, except for *Vulnerability (consequence)* column which has been removed as the concept of vulnerability in the context of fairness does not apply.

The sheet containing the knowledge for "**Explainability**" has a different set of columns (see Fig. 4.17), because there are no real threats associated with the lack of explainability. However, having a system that is not explainable, will lead users to use it with some reluctance because of its opacity in making decisions, as it is not possible to derive any clear logical relationship

between the internal configuration and their external behaviour, except for a few specific cases (e.g. decision trees) [317].

For Explainability, the columns are the following:

Figure 4.17: Excerpt of the Explainability component navigation of the POLARIS framework. The whole content can be found in the online appendix [10].

SDLC Phase	Data type	Local/Global Explanation	Explanation Goal	Action
RE	General	Both	Start considering Explainability from Req. Elicitation	Elicit also explainability requirements; examples are: - Unexpected Prediction: Disagreement with AI: declare the required behaviour in case the AI prediction is unexpected, and/or users disagree with AI's prediction - Expected prediction: declare the required behaviour in case AI's prediction aligns with users' expectations - Differentiate similar instances: due to the consequences of wrong decisions, users sometimes need to discern similar instances or outcomes. For example, a doctor differentiates whether the diagnosis is a benign or malignant tumor - Learn from AI: users need to gain knowledge, improve their problem-solving skills, and discover new knowledge - Improve the predicted outcome: users seek causal factors to control and improve the predicted outcome - Communicate with stakeholders: many critical decision-making processes involve multiple stakeholders, and users need to discuss the decision with them - Generate reports: users need to utilize the explanations to perform particular tasks such as report production. For example, a radiologist generates a medical report on a patient's X-ray image
Design	General	Both	Have a clear idea about the desired explanation form	Consider the design of the explanation design: how the final UI should be composed and how to present the information

(1) SDLC Phase. The SDLC phase the *Action* column relates to.

(2) Data Type. The type of data used by the AI algorithm for which the action/guideline applies. Examples are *Tabular* data or *Image*. When the action applies to all algorithms, regardless of the type of data, the tag *General* is used.

(3) Local/Global Explanation. This column describes the type of explanation that can be obtained by implementing the action. At the moment, the possible values are *Global* and *Local* [318].

(4) Explanation Goal. This is the goal that can be achieved if the action/guideline gets implemented. Examples are: *to validate the algorithm outcome* and *to reveal bias*.

(5) Action. The corresponding action, or decision, that should be taken to reach the selected explanation goal. We point out that for each <data, explanation type> pair there is at least a corresponding row in the framework.

For example, a user who is in the *Requirement Elicitation* SDLC phase and needs to enquire on all the possible explanation approaches to explain the output of an algorithm, could access the *Explainability* Excel sheet and select the *General* data type and retrieve a set of actions that pertain explainability requirements i.e. *elicit explainability requirements* (Fig. 4.17).

When navigating POLARIS, each stakeholder can use different filters and subfilters, based on specific needs, as for instance: *Knowledge Component* (i.e. TAI principle) to address, *Threat* (or *Sub-Threat*), *Vulnerability*, *SDLC Phase*, *Data type*, and *Local/Global Explanation*.

One of the most significant filters is *SDLC phase*, which makes POLARIS flexible and allows stakeholders to use it either on ongoing/closed projects — where it is possible to address, for example, only the deployment or monitoring phase — or at the early stage of a project, since in the latter case it can cover all SDLC phases.

4.3.3 Application to real contexts

With the aim of validating POLARIS framework and investigating key points for improvement, I have applied the framework in an industrial project and collected feedback from practitioners through a think-aloud session as they applied the framework during the development of an AI-enabled application. The following sections give more details about the industrial case, the improvements made after having collected feedback and the lessons learnt from applying the framework to a first industrial real case.

Industrial Case setting

I applied POLARIS during the development of a real-world software application on behalf of a SME. In accordance with the guidelines provided by Runeson and Höst [319], the **research question (RQ)** I aimed to address questioned: *Is POLARIS helpful in supporting different stakeholders dealing with Trustworthy AI challenges in all the SDLC phases?* With this RQ in mind, I focused on how and to what extent POLARIS helps stakeholders. Furthermore, I take the feedback received to improve the framework itself. During the development of the application, developers used POLARIS and were, at the same time, involved in think-aloud sessions to collect feedback.

The **case selection** involved an AI-enabled software application developed by an IT SME that mainly provides services in various domains such as cybersecurity, AI, and Software Engineering. The IT team of the company is composed of 7 seniors, 3 mid-experienced, and 15 juniors.

The application was commissioned to the company by a public entity that was particularly committed to AI ethics and trustworthiness issues. The software application had to perform image recognition and object detection tasks. Moreover, since the application is used in a very sensitive and critical domain, the system could not make autonomous decisions but had to only provide suggestions to the human operator — functioning as a decision support system. One of the requirements was that each suggestion must be supported by a graphical, self-explainable and human-readable description. This application required dealing with *computer vision*, specifically *image analysis*, so includes methods for acquiring, processing, analyzing, and understanding digital images [320].

The IT teams chose to use a deep-learning model to perform these tasks. This meant that several images were necessary to conduct a proper training phase and achieve high-level performances. Since training data was not available, except for only a few samples, the developers artificially created a dataset starting from the images available in four open-source datasets.

Due to the sensitivity of the domain, the entire system was deployed on an on-premise infrastructure.

While designing and developing the application, developers applied the POLARIS framework components during all phases of the SDLC.

The **data collection** (3) was conducted by means of the concurrent thinking-aloud method [321, 322]: I first explained to a software development team how POLARIS was structured and should be integrated into their work; then, I participated in their daily routines (on days agreed upon with them) and asked them to think aloud as they used POLARIS during all moments of the SDLC. The team was composed of 1 Business Analyst (**BA**) — who tested POLARIS Fairness and Explainability components during the entire Requirement Elicitation phase — 1 Senior Software and Data Engineer (**SDE**) — who tested Fairness and Explainability during the Design and Development phases — 1 Junior Developer (**DEV**) — who tested Fairness and Explainability during the entire Development phase — 1 Cybersecurity Specialist & DevOps (**CS**) — who tested Privacy and Security during the entire SDLC — Quality Assurance Engineer (**QA**) — who tested Fairness and Explainability during the entire Test phase — and 1 IT engineer (**IT**) — who tested Fairness and Explainability during the Deploy and Monitoring phases. During sessions, I took notes. In particular, the researcher reminded participants to think aloud and solicited comments through questions to motivate feedback.

The **data analysis** (4) was conducted by means of qualitative methods. Once the software system was delivered to the final customer, I analyzed all the collected feedback notes to conclude to what extent POLARIS supports managing TAI principles and establish areas of improvement. The main outcomes of the study along with the improvements applied to the framework, are reported in Section 4.3.4.

As recommended by Creswell [323], to ensure the **validity of the findings**, I followed the *member check* strategy and asked all team members involved in the study to validate my findings and provide feedback on the sessions.

4.3.4 POLARIS evolution

Once the study was completed, I used the results collected from the feedback sessions to implement a set of changes intended to improve POLARIS.

In this section, I outline the most important findings that have led to changes made to POLARIS and explain the rationale behind each one. The complete list of changes can be found in the online appendix [10].

Adding filtering capabilities. The initial POLARIS version did not have any filtering options. It was a single Excel file, thus the users must vertically scroll the entire sheet to find the relevant information for a specific component or (sub-)threat. Moreover, there was a column for each SDLC phase.

However, all team members reported that POLARIS was very cumbersome and difficult to use with these settings. So, the first improvements were related to usability: I created a single file for each TAI principle and inserted a dropdown list allowing to to choose the SDLC phase of interest. Other filtering options have also been applied. For space reasons, I have not detailed them.

Revise the mapping between SDLC phase and guidelines. SDE, DEV, CS, QA and IT reported that it was necessary to review the SDLC phase with which some guidelines were mapped. In particular, there were guidelines (e.g., "*Implement processes to maintain security levels of ML components over time*" and "*Include ML applications into detection and response to security incident processes*") that were too abstract to be mapped to practical SDLC phases (e.g., *Development* and *Test*).

Conversely, some guidelines were mapped only with the later phases of the SDLC, while my testers highlighted the need for help early on in the process. For example, the guideline "*Define and monitor indicators for proper functioning of the model*", according to ENISA, was mapped only with *Monitoring* phase; I also added the *Design* phase because it is important to start defining the *Key Performance Indicators* (KPIs) even before starting the development of the model itself: once the model is developed, missing metrics can be discovered but modifying the code may now cost too much time and/or money.

Remove TAI patterns. Initially, I included all TAI patterns provided in Data61-CSIRO's "*Responsible AI Pattern Catalogue*" [314] in POLARIS, because I assumed every pattern could help developers to incorporate TAI in the development phase. Feedback from SDE, DEV, and CS, when applying these patterns, pointed out that some patterns were not really applicable: for example, the pattern "*TAI user story*" gets implemented simply by using POLARIS itself. I revised and removed the patterns that were not considered relevant or useful.

Remove *governance-oriented* guidelines. All team members pointed out that there were guidelines that cannot really be implemented through the SDLC phases, but require an organizational posture change, with new aspects to be considered in their internal policies. This happened, for example, for the guideline "*Integrate ML specificities to awareness strategy and ensure all ML stakeholders are receiving it*". So, I decided to remove such guidelines from POLARIS because the framework is not meant to provide suggestions at the organization's policy level.

Remove equivalents guidelines. Since I included guidelines from different knowledge sources, I initially missed the fact that diverse guidelines may address the same (sub-)threat and provide the same mitigation but with different words. After SDE, DEV, CS, QA, and IT reported this situation, I investigated and removed these redundancies. For example, Microsoft's guideline "*Input validation, both sanitization and integrity checking*" was deleted because almost identical to the ENISA's "*Control all data used by the ML model*".

Highlight the fact that some guidelines provide different alternative mitigations for the same threat. Especially in the Development and Test phases of the SDLC, there are a lot of alternative tools to automatically address a specific threat. SDE, DEV, IT, and QA pointed out that a less familiar stakeholder may miss this nuance and implement all suggestions, discovering only in the end that all of them led to the same result. So, I decided to make clearer the cases where the user can simply choose one alternative instead of applying the entire array of mitigations.

Contextualize abstract guidelines. Analyzing the different knowledge sources, SDE and CS found that some guidelines sound too abstract to be brought back to something concretely implementable. I reviewed these cases, analyzing the literature to find the best actionable mitigations, and integrated them into POLARIS. For example, this process was conducted for the ENISA's guideline "*Choose and define a more resilient model design*": here I added more specific and concrete examples by integrating the knowledge provided in [324].

Merge similar guidelines. In the Security component — where I included the guidelines proposed by both Microsoft [316] and ENISA [312] — CS found that most of the guidelines proposed by Microsoft addressed the same threat already addressed by ENISA; in some cases, Microsoft proposed one or few additional actions compared to ENISA (or vice-versa). To address this issue, I decided to take the ENISA's guideline — because was the most comprehensive — and merge into it the missing details from Microsoft's guideline. For example, this happened for the "*Lack of training based on adversarial attacks*" vulnerability, linked to *Evasion* threat.

Add missing tools. While using the framework, DEV, CS, QA, and IT reported to us that they found some guidelines that were mainly descriptive and did not provide any tools useful for implementing those guidelines. Since in many cases, they knew there was a tool to support mitigation strategies, I added all the tools they suggested. For example, I added ENISA's online tool for the security of personal data processing[24] for conducting DPIA and support the guideline "*ICO 1.1 - Conduct a data protection impact assessment (DPIA)*".

Remove poorly documented concepts. BA and SDE pointed out that some guidelines were too abstract to be implemented. After analyzing the above guidelines and finding that the current literature did not provide enough information to make the guideline actionable, I decided to remove this category of guidelines. For example, in the Fairness component, I removed the "*construct validity*" concept because it is poorly documented and rarely mentioned in the current literature; additionally, there are no specific indications of what mitigations exist and how they can actually have a practical impact through the SDLC.

Lessons learnt

In addition to the improvements discussed in Section 4.3.4, thanks to the case study, I identified additional insights that may be useful to further improve POLARIS framework and its usability.

In this section, I describe some lessons learnt from the analysis of the case study data.

User Experience (UX). From the testing of POLARIS what clearly emerged is that an Excel sheet provides a suboptimal user experience, and the resulting complexity of use can lead to new users abandoning POLARIS. For this reason, then I planned to provide a better UI to make POLARIS consultation easier and faster.

To achieve this, I evaluated various types of UI: a questionnaire-like web page, a Single Page Application (SPA) — like the VIS-Prise tool proposed in [229] —, or a book-like web page —

[24] `https://www.enisa.europa.eu/risk-level-tool/risk`

like the PAI's Guidance for Safe Foundation Model Deployment[25]. All the details are described in the next chapter.

Lack of concrete guidelines for the Monitoring phase. While observing the IT user utilizing POLARIS, I noticed that the monitoring SDLC phase is the one with the least concrete guidelines, probably due to the fact that the advent of the cloud has, in some way, delegated the monitoring burden to third parties. Anyway, since the IT user had the necessity to deploy an AI-enabled system on an on-premise infrastructure, he pointed us to this lack of the framework. For this reason, I plan to conduct further research in the current literature to collect as many concrete monitoring guidelines as possible; this way, POLARIS would become more effective in the monitoring phase.

Lack of specific tools and guidelines for LLM. Most of the knowledge sources that compose POLARIS were published before the recent great improvements in Large Language Models (LLMs). As a consequence, POLARIS now lacks guidelines and tools specific to LLMs. As all team members pointed out, LLMs are becoming increasingly used in AI-enabled systems and they plan to use them in in the next version of the application on which I tested POLARIS. For this reason, I plan to research, integrate and experiment with both guidelines and automatic tools to specifically address trustworthiness in LLMs.

Monitor literature advancements. POLARIS for its nature is a live framework that needs to be regularly updated as the literature in the AI-enabled systems field is constantly evolving. We continuously monitor new grey- and white-literature sources. This way, I can timely discover new knowledge sources (both guidelines and tools) that can be integrated into POLARIS and keep it updated. For example, I am evaluating the new MITRE Atlas mitigations collection[26].

Make the framework open source. The reflections on the previous point lead us to the idea that POLARIS framework should be published on an open-source repository, freely accessible under the CC-BY 4.0 license, so that anyone interested in the project can integrate new knowledge or refine the existing one. This way, I could exploit the wisdom of the crowd to further enhance POLARIS. For this reason, I published POLARIS in a public GitHub repository [325].

4.3.5 Empirical Study

The last part of my research was the validation of the proposed POLARIS framework, which is also described in a paper currently submitted. To this aim, I first realized a new version of the UI and then conducted focus groups with the employees of 2 companies. During these focus groups, the participants had the opportunity to be instructed on the TAI theme and on the POLARIS knowledge base, as well as to use POLARIS to solve some proposed tasks.

[25] `https://partnershiponai.org/modeldeployment/`
[26] `https://atlas.mitre.org/mitigations`

At the end, they were asked to fill out the Unified Theory of Acceptance and Use of Technology (UTAUT) questionnaire, in order to measure their perceived *System Acceptance* and *Perceived Trust in the System*.

Elicitation study and first UI draft

To start validating POLARIS with final users, I needed to realize a web application with all the information initially provided as an Excel file. I aimed to realize this web application so that it could reflect the users' requests. An elicitation study was carried out to design a web application that users may use effectively. To that purpose, a five-step methodology was followed in accordance with the instructions of [326] to collect the requirements.

Participants

For this study, I recruited practitioners working on AI products and services as well as front-end web developers through a combination of purposive and snowball sampling [293]. The target participants for the survey were (a) AI practitioners involved in the development of AI-based systems, preferably with some basic knowledge of TAI principles and/or who had previously addressed TAI in their professional work, and (b) front-end web developers with knowledge about UI and UX concepts. In this study, I involved employees from 2 private companies that had for sure the required technical skills.

Overall, I interviewed 5 web developers, 1 UI/UX Designer and 1 Data Scientist.

Procedure

We conducted a series of semi-structured interviews (synchronous interaction). All participants answered the same set of questions and had the opportunity to add any non-listed suggestions/feedback related to the closed-ended question in the open-text fields (asynchronous mode), as well as "discuss their answers out loud" and further elaborate their considerations with the interviewers. All sessions were held in an office environment familiar to the participants. To be included later in the thematic analysis, the answers were all transcribed via Microsoft Teams[27]; after asking each participant for their consent, we we shared the screen and then enabled *Recording & Transcription* Teams features. Once the interview ended, we proof-checked the transcription in order to correct any misspellings, anonymized any Personal Identifiable Information (PII) and finally deleted the recording. Participation was on a voluntary basis and not rewarded by any means. The interview canvas contains six main sections and can be accessed at [327]:

1. *Preliminary concepts knowledge.* I clarified the semantics and interpretation of each TAI principle for participants by providing a definition for each principle. By listing a definition, I wanted to build a shared understanding of each principle to answer the remaining questions. In addition, I asked the participants about their vision and previous experience with TAI.

2. *How to.* I asked their opinion about the presence of a section in which I explained how to use POLARIS UI.

[27] `https://www.microsoft.com/it-it/microsoft-teams`

3. *Home*. I asked their opinion about what to insert on the homepage.

4. *POLARIS Consultation*. I asked their opinion about what to insert on the homepage.

5. *Support & Collaboration*. I asked their opinion about how useful they consider a section where they can cooperate with other practitioners (e.g. a forum).

6. *Updates & News*. I asked their opinion about how useful they consider the possibility of receiving updates on POLARIS evolution.

7. *UX & What to avoid*. I asked for suggestions about common UI/UX mistakes they would like to highlight.

8. *Accessibility & Ease of use*. I asked for suggestions about common accessibility mistakes they would like to highlight.

9. *Access Devices*. I asked about the kind of access device they plan to use (e.g. mobile device or notebook).

10. *Demographics and background information*. Participants were asked about gender, level of education, country, role, years of experience, size of the organization, etc.

The survey was anonymous and did not ask for any directly identifying information. Most of the survey questions were closed-answer and mandatory, but there were also optional open-text ones. Through the latter, I was able to collect further qualitative data as many of the participants provided information on the practices they usually implement. All survey data and raw material can be accessed in the online appendix [328].

Before scheduling the interviews with participants, to understand the challenges and requirements for conducting remotely semi-structured interviews, as well as to help us refine this study protocol, I conducted two pilot interviews. These preliminary interviews helped us define the protocol and the setting I applied to the final, larger sample. All interviews were conducted in person. In the end, the official interview study consisted of think-aloud semi-structured interviews, each one lasting a minimum of 20 minutes and a maximum of 40 minutes. During the live interview, I periodically asked participants to elaborate on their responses, especially for the open ones. I also encouraged participants to "think aloud" [296, 297] and discuss the information that was being displayed and how their understanding of the question was developing.

Materials

In this step, I extracted all relevant data using quantitative and qualitative data analysis techniques to summarize and interpret the collected data. For quantitative data, I used descriptive statistics [298], and for qualitative data, I used thematic analysis [299]. We used an **inductive thematic analysis** approach [300, 301] to analyze all the gathered data. The entire analysis was done through Atlas.ti[28]. Two researchers worked independently and used the tool to conduct an open coding of the transcripts for each quotation. Next, they manually reviewed each code and decided which to include/exclude annotating any comments. Once this step was completed, they joined to compare and discuss results. The total number of analyzed quotations was 55. All the details about the coding procedure and the generated codes are

[28] `https://atlasti.com/`

provided in the online appendix [328]. Once finalized, the codes were shared with the entire research team and grouped into higher-level themes concerning the practitioners' knowledge and practices. In next section, I discuss the findings identified from these codes and themes, together with implications for future TAI developments.

Results and findings

In this section, the findings from the data analysis are presented as a list of requirements:

- Training knowledge in the form of PDF and video
- After each training section, provide quizzes with hints in case of errors and non-blocking for navigation
- Provide TAI training in a separate website page and, also, as a tooltip when mousing in the actions filtering section
- Provide a search function via a filtering button and a search bar
- Chatbot assistance is considered superfluous in many cases; users suggested implementing it only via an LLM and not the classic chatbot with predefined answers
- The presence of a collaborative forum presence is fine but it must be moderated and there must be someone who always tries to reply
- A section to contribute to POLARIS development should be comprised of both an issue notification system and a ticketing service to propose enhancements
- A newsletter would be useful only in the case of substantial upgrades and not more often than once a week
- Regarding the layout, users asked for responsiveness, compliance with material design, a navbar that does not disappear when scrolling

Graphical result

Fig. 4.18 shows the first UI version I realized.

UI revision

In this research step, I asked for help from a UX designer with more than 5 years of job experience. This person inspected the first POLARIS web interface version and found some critical points to be addressed:

- *Logo too complex*: the elements that compose it must be simpler and be balanced in terms of visual weight
- A *slogan* that introduces POLARIS is *missing*

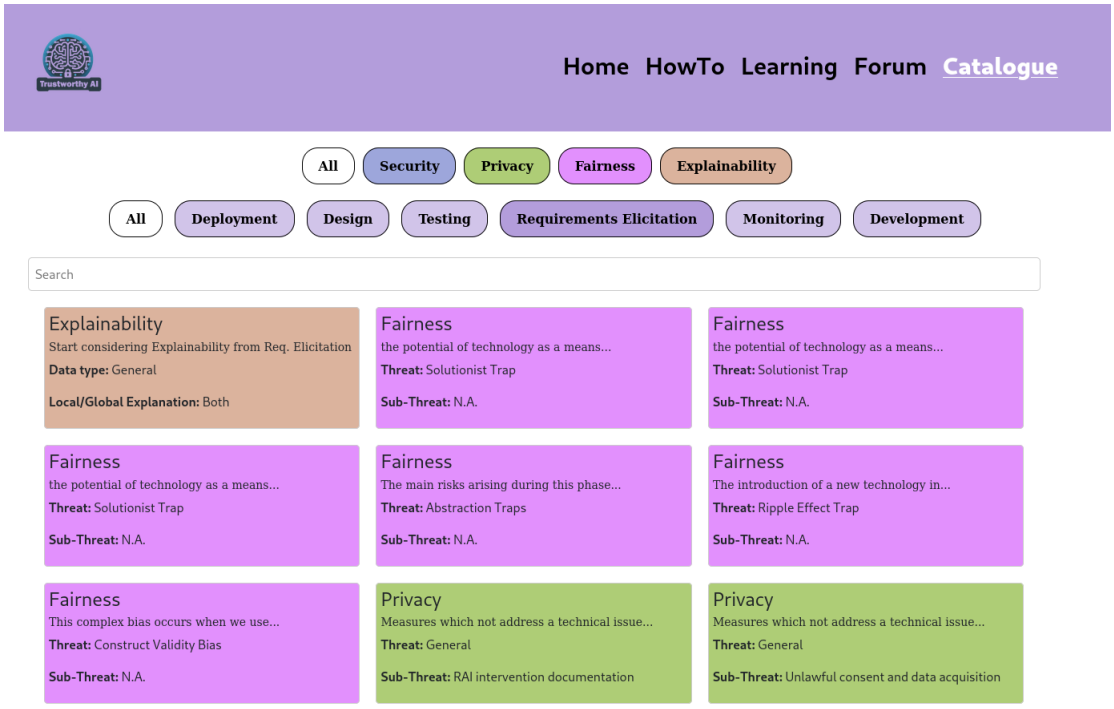


Figure 4.18: First POLARIS web UI version.

- Some *elements* in the homepage, like the TAI pillars descriptions, have *no context*
- *Too much text* is provided in the homepage
- The *icons* have *not the same visual format* (they differ in terms of font size and weight)
- There are *no subtitles* in the tutorial videos
- The *filtering experience must be revised*: it is not clear if the buttons' order is important and what can be expected from interacting with them
- The *layout* of the *Training* section must be changed and the information must be *reorganized* because it is too difficult to find the important information
- The *supplementary material* associated with a *guideline* in the catalogue must be provided at the end of the card

Later, to validate these preliminary findings, I developed new digital prototypes and conducted user interviews in which I asked 5 AI practitioners — who represent potential users for POLARIS — to do 3 tasks on these new digital prototypes. Fig. 4.19 shows an excerpt of the new prototypes I used in the interviews.

Materials

Since the amount of data was smaller than in the previous user study, the UX expert handled solo the data validation and provided me with a summary of the results. The report was split into two macro-sections: *Things that worked well* and *Things that do NOT worked well*.

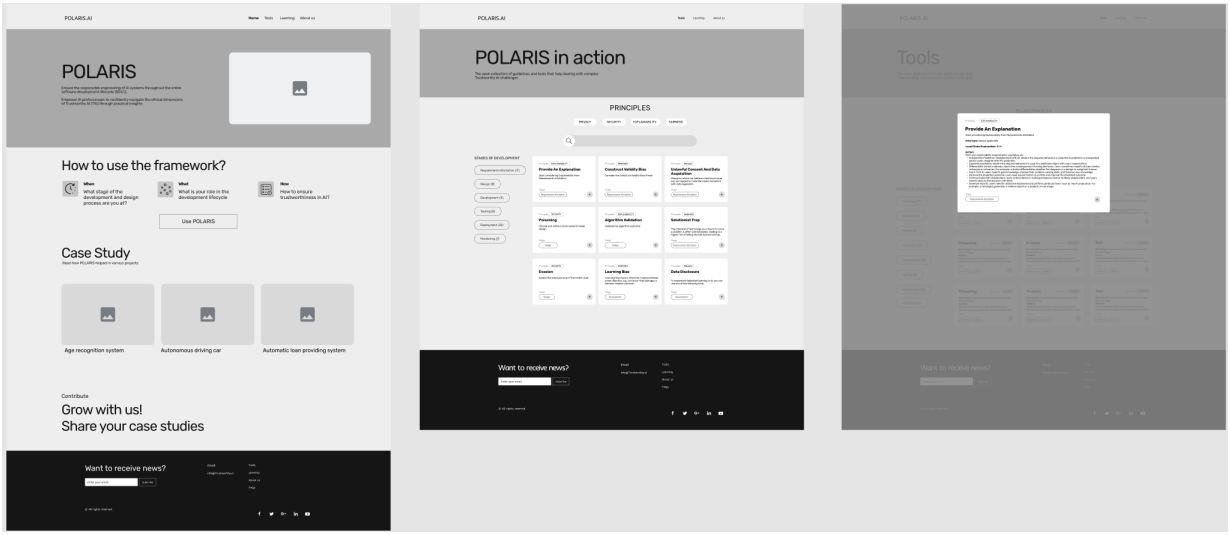


Figure 4.19: New POLARIS web UI prototypes.

Results and findings

In this section, the findings from the user interviews study are presented as a list of key points.

In section *Things that worked well*, the key findings are:

- *Clarity of the information*: cards were brief and provided key information quickly
- *Clear filters and navigation*: users appreciated that the filters allowed information to be segmented according to the stage of the project in which they were interested
- *Specific information on key topics*: topics such as privacy and data security were well covered and easily found by participants

In section *Things that do NOT worked well*, the key findings are:

- *Lack of detail on required actions*: the information provided was considered good but brief, lacking a level of detail regarding the actions to be taken to ensure compliance
- *Initial confusion with filters and interaction elements*: initially, some users did not understand that the filters on the left and those at the top corresponded to different concepts

After this analysis, I realized the second version of the Polaris UI. Fig. 4.20 shows an excerpt of this second UI version.

This UI was realized as a web application, that can be navigated at `http://193.204.187.41:8001`.

POLARIS Final User Study

In this section, I describe the first round of user interviews conducted to validate the Polaris knowledge base, as well as the results obtained from the thematic analysis and the USE and

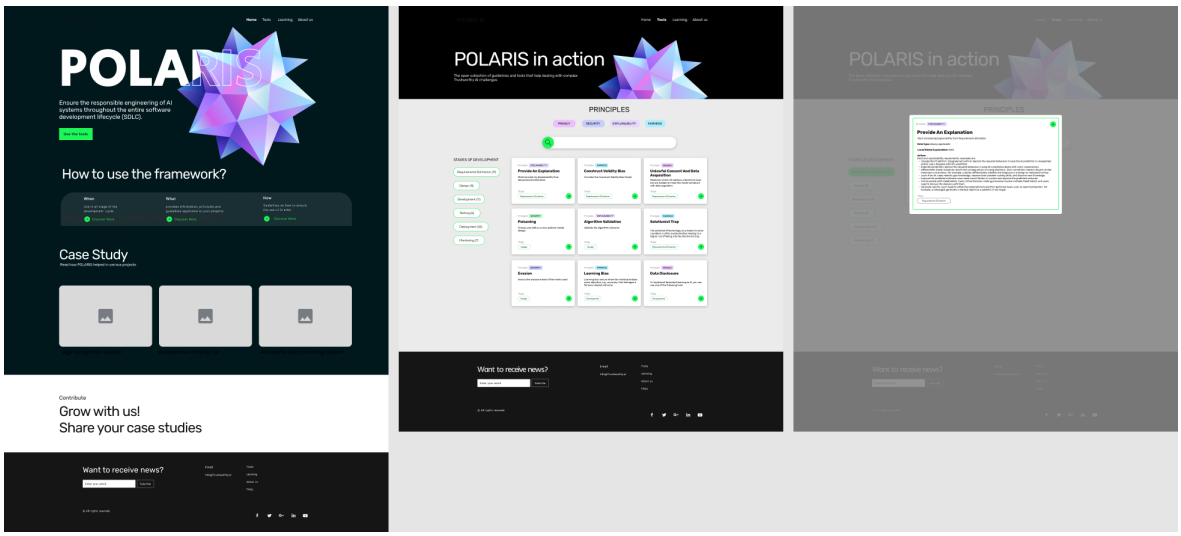


Figure 4.20: POLARI UI second version.

TAM questionnaires.

This initial study consisted of 6 interviews, but I already planned to extend the study: I planned to organize focus groups with different companies of different sizes, to obtain useful different insights and further enhance Polaris.

Participants

For this study, I recruited practitioners working on AI products and services from one big software company in Bari. The target participants for the survey were AI practitioners involved in the development of AI-based systems, preferably with some basic knowledge of TAI principles and/or who had previously addressed TAI in their professional work, not involved in any previous POLARIS validation.

Procedure

I conducted a series of semi-structured interviews (synchronous interaction). All participants did the same usability tasks and answered the same questionnaires (USE and TAM) and had the opportunity to "discuss their answers out loud" and further elaborate their considerations with the interviewers. To be included later in the thematic analysis, the answers were all transcribed via Microsoft Teams[29]; after asking each participant for their consent, we enabled *Recording & Transcription* Teams features. Once the interview ended, we proof-checked the transcription in order to correct any misspellings, anonymized any Personal Identifiable Information (PII) and finally deleted the recording. Participation was on a voluntary basis and not rewarded by any means. The survey can be accessed at [329].

The interview canvas contains six main sections [330]:

1. *Preliminary concepts knowledge*. I clarified the semantics and interpretation of each TAI principle for participants by providing a definition for each principle. By listing a definition, I wanted to build a shared understanding of each principle to answer the remaining questions. In addition, I asked the participants about their vision and previous experience with TAI.

[29] `https://www.microsoft.com/it-it/microsoft-teams`

2. *Context.* I narrated a realistic work context, with possible TAI problems, in which the interviewee had to empathise in order to solve the tasks.

3. *Demo Task.* I analyzed with the interviewee a task and then showed how to solve the task.

4. *Task 1.* We asked the interviewee to solve a task with complete autonomy. The goal was to find the appropriate guidelines to avoid the *Unlawful consent and data acquisition* problem.

5. *Task 2.* We asked the interviewee to solve another task with complete autonomy. The goal was to find a software library that *generates artificial samples.*

6. *Questionnaires and Feedback.* I asked the interviewee to fill out the USE and TAM questionnaire, as well as provide general demographic questions like gender, level of education, country, role, years of experience, size of the organization, etc.

The survey was anonymous and did not ask for any directly identifying information. All survey data and raw material is provided in the online appendix [330].
Before scheduling the interviews with participants, to understand the challenges and requirements for conducting remotely semi-structured interviews, as well as to help us refine this study protocol, I conducted two pilot interviews. These preliminary interviews helped us define the protocol and the setting I applied to the final, larger sample. All interviews were conducted in person. In the end, the official interview study consisted of think-aloud semi-structured interviews, each one lasting a minimum of 30 minutes and a maximum of 60 minutes. During the live interview, I periodically asked participants to elaborate on their responses, especially for the open ones. I also encouraged participants to "think aloud" [296, 297] and discuss the information that was being displayed and how their understanding of the question was developing. To give a standard structure to each interview, I used an interview canvas [330].

Materials

In this step, quantitative and qualitative data were collected. Regarding quantitative data, *task success rate* and *task time* were measured to determine the effectiveness and efficiency of the participants in completing the given tasks. In addition, the USE (Usefulness, Satisfaction, and Ease of Use) questionnaire, a standardized assessment tool for measuring the usability and user experience of a product, was administered [331] together with TAM (Technology Acceptance Model) questionnaire [332], used to investigate if users accept or reject a new technology. For qualitative data, observational notes were taken during the study.

As in the elicitation study, for quantitative data, I used descriptive statistics [298], and for qualitative data, I used thematic analysis [299]. I used an **inductive thematic analysis** approach [300, 301] to analyze all the gathered data. The entire analysis was done through Atlas.ti[30]. Two researchers worked independently and used the tool to conduct an open coding of the transcripts for each quotation. Next, they manually reviewed each code and decided which to include/exclude annotating any comments. Once this step was completed, they joined to compare and discuss results. The total number of interesting and analyzed citations was

[30] `https://atlasti.com/`

Table 4.8: Overall quantitative results of the user study.

Participant	Task 1 Outcome	Task 1 Duration (s)	Task 2 Outcome	Task 2 Duration (s)
1	KO	455	OK	125
2	OK	110	OK	175
3	OK	174	OK	64
4	OK	110	OK	110
5	OK	154	OK	90
6	OK	80	OK	144
Overall	83.33%	3min and 05s	100%	2min and 58s

14. All the details about the coding procedure and the generated codes are provided in the online appendix [330]. Once finalized, the codes were shared with the entire research team and grouped into higher-level themes concerning the practitioners' knowledge and practices. In next section, I discuss the findings identified from these codes and themes, as well as the results from the USE and TAM questionnaires.

Results and findings

The analysis of the data collected during the study starts by considering the task success rate and the execution times; all details are in the online appendix [330]. Tasks were considered failed if the final goal was not successfully reached, or if participants took a path they were not supposed to take. In terms of time, I considered 180 seconds as the failure threshold. Table 4.8 shows the overall results.

Task 1 was executed with few errors (83,33% success rate), while Task 2 had 100% success rate. Regarding the time, participants spent respectively 181 sec and 118 sec to complete the tasks. Task 2 was completed in less time because participants spent more time in Task 1 to familiarize themselves with the UI, while in Task they were already familiar with the environment. Analyzing the video of Task 1 failed by P1, it emerged that the participant failed the task since he followed the wrong path to reach the goal, choosing the wrong SDLC phase from the lateral checkbox.

The analysis of the USE questionnaire has been conducted along its four dimensions, i.e., usefulness, ease of use, ease of learning, and satisfaction. Regarding *usefulness* (see Figure 4.21), it resulted in an average score of 5.4 out of 7, which indicates that users find the web platform to be very useful. This is a positive result, showing that the platform is meeting users' needs effectively.

Looking in detail at the eight questions of this dimension (see Figure 4.22), it is evident that Q4 (*It gives me more control over the activities in my job.*) and Q8 (*It does everything I would expect it to do.*) achieved the lowest score, despite still obtaining positive results; these scores indicate that I need to make it clearer that POLARIS does not automate the mental process behind the operative decisions in software projects and that POLARIS provides only decisional support and does not substitute the human decision. On the contrary, Q3 (*It is useful*) highlights that users perceived the platform as very useful in their work.

In terms of *ease of use*, the platform is considered fairly easy to use, with a score of 5.9 out of 7. Although this is not a perfect score, it is still high and suggests that users generally find the

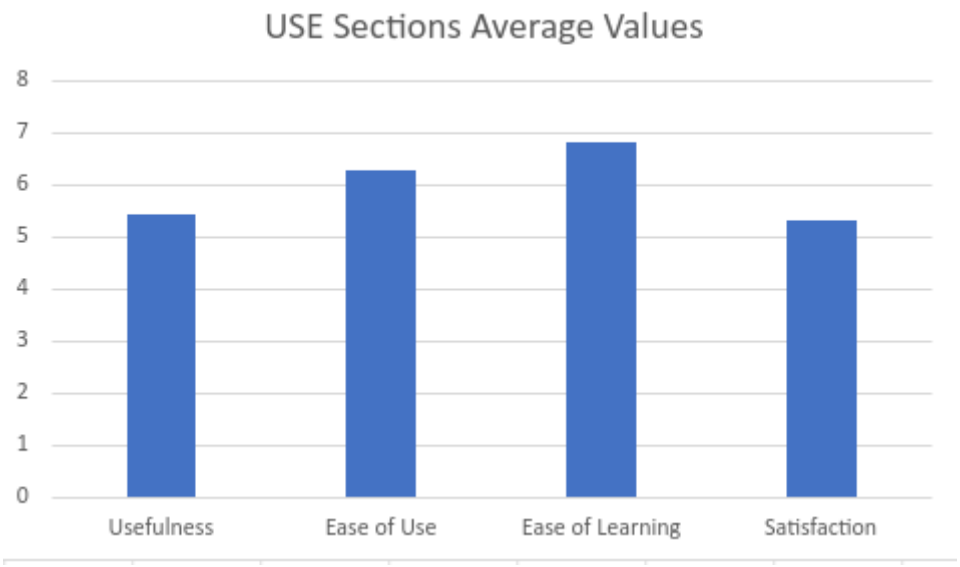


Figure 4.21: USE usefulness results.

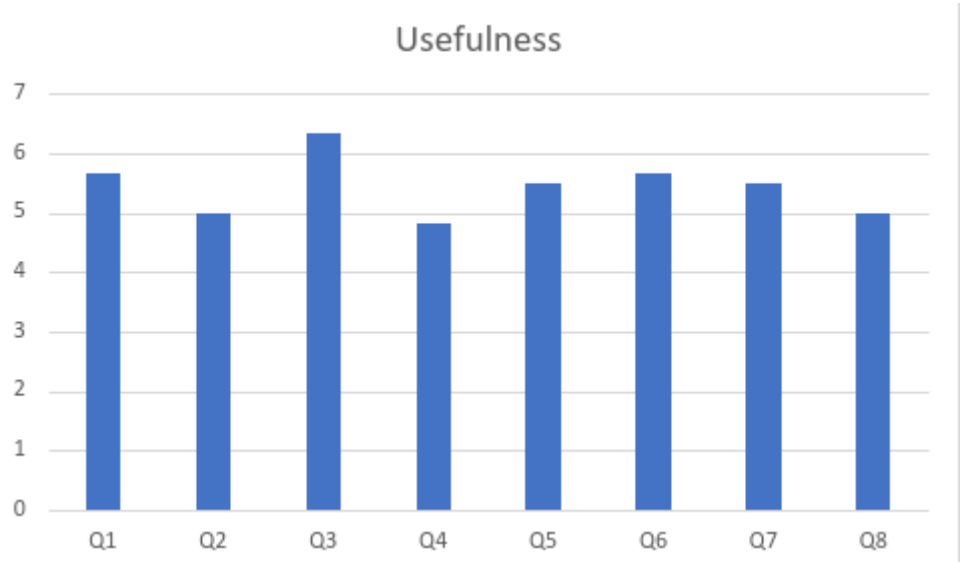


Figure 4.22: USE usefulness detailed questions results.

platform easy to use, while there may be some room for improvement in terms of usability. As shown in Figure 4.23, the questionnaire items that most negatively affected this score are Q13 (*It is flexible.*): this is due to the fact that the knowledge base is static and not automatically updates, so they may sometimes find obsolete information.

Regarding the *ease of learning*, the achieved score of 6.8 out of 7 suggests that users find the platform very easy to learn, indicating that users can quickly grasp how to use the platform and that there is a low learning curve. The details reported in Figure 4.24 clearly show that all questionnaire items in this dimension contributed in a fairly equal way to the final score, with no particular positive or negative choices.

Finally, the *satisfaction* dimension, with a score of 5.3 out of 7, indicates a high level of satisfaction among users, which is a positive sign that they are satisfied with their overall

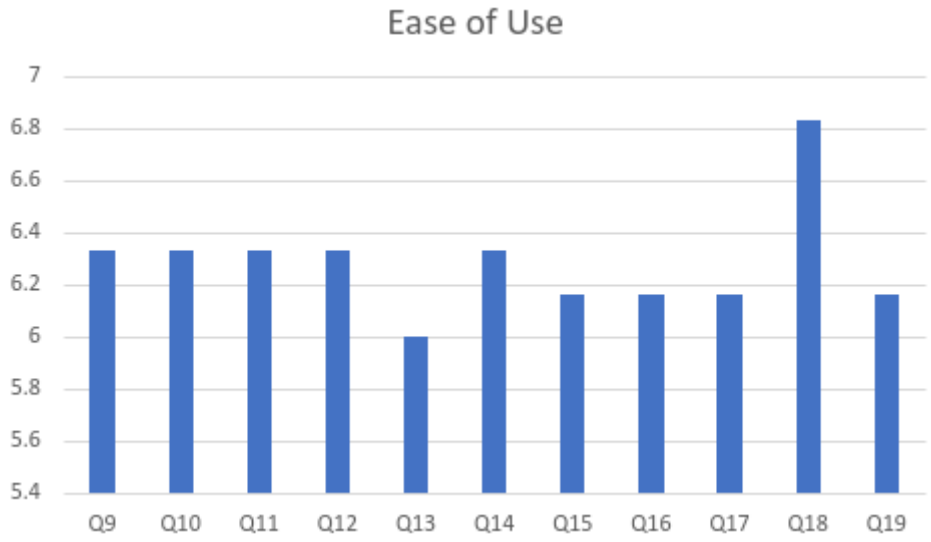


Figure 4.23: USE ease of use detailed questions results.

Figure 4.24: USE ease of use learning questions results.

experience. The details depicted in Figure 4.25 indicate that only two items in this dimension might deserve attention, i.e., Q29 (*It is fun to use.*) and Q30 (*I feel I need to have it.*). While Q29 can be associated with the hedonic aspect of the platform, which is not critical to its purpose, more interesting evidence may come from Q30, as it reveals the need to better adapt the platform to the user's needs.

The analysis of the TAM questionnaire has been conducted along its two dimensions, i.e., perceived usefulness, and perceived ease of use. Regarding *perceived usefulness* (see Figure 4.26), it resulted in an average score of 5.5 out of 7, which also indicates that users find the web platform to be very useful.

Looking in detail at the six statements of this dimension (see Figure 4.27), it is evident that Q3 (*Using this product would increase my productivity.*) achieved the lowest score, despite still

Figure 4.25: USE satisfaction questions results.

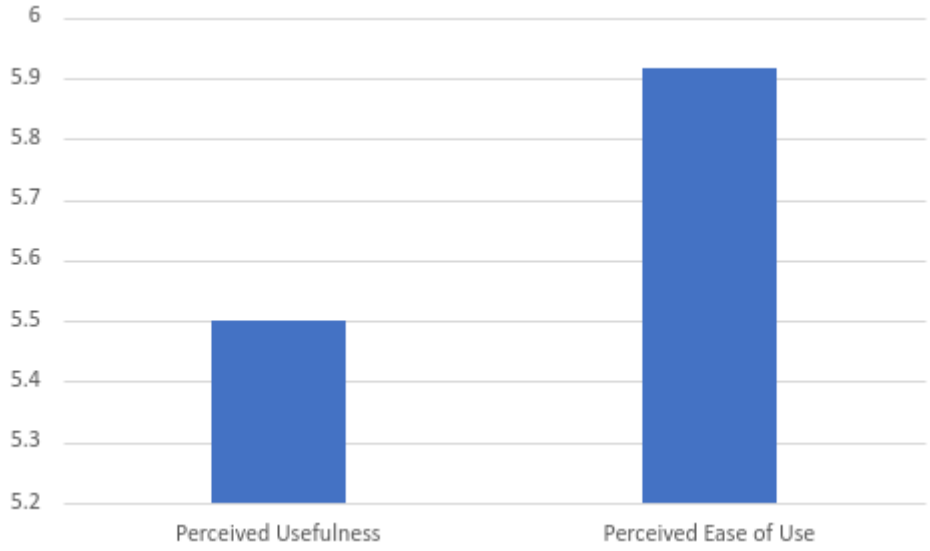


Figure 4.26: TAM overall results.

obtaining positive results; these scores indicate that I need to make it clearer how POLARIS can increase the user productivity even if it does not automate the mental process behind the operative decisions in software projects. On the contrary, Q4 (*Using this product would increase my effectiveness at work.*) highlights that users perceived the platform as very effective during their daily work.

In terms of *perceived ease of use*, the platform is considered fairly easy to use, with a score of 5.9 out of 7. Again, although this is not a perfect score, it is still high and suggests that users generally find the platform easy to use, while there may be some room for improvement in terms of usability. As shown in Figure 4.28, the questionnaire item that most negatively affected this score is Q11 (*It would be easy for me to become agile with the product.*): this is due to the fact that some cards are rich of text, so users may be scared of wasting time by

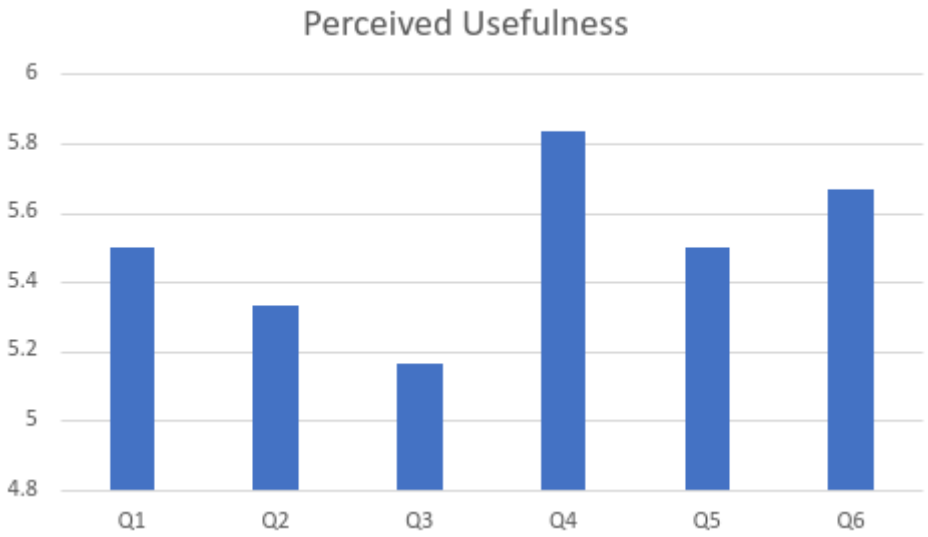


Figure 4.27: TAM perceived usefulness detailed questions results.

reading the whole text. On the contrary, the questionnaire item that most positively affected this score is Q8 (*I would find it easy to let the product do what I want it to.*): this highlights how POLARIS is easy to use and the user can fastly grasp how to make it work.

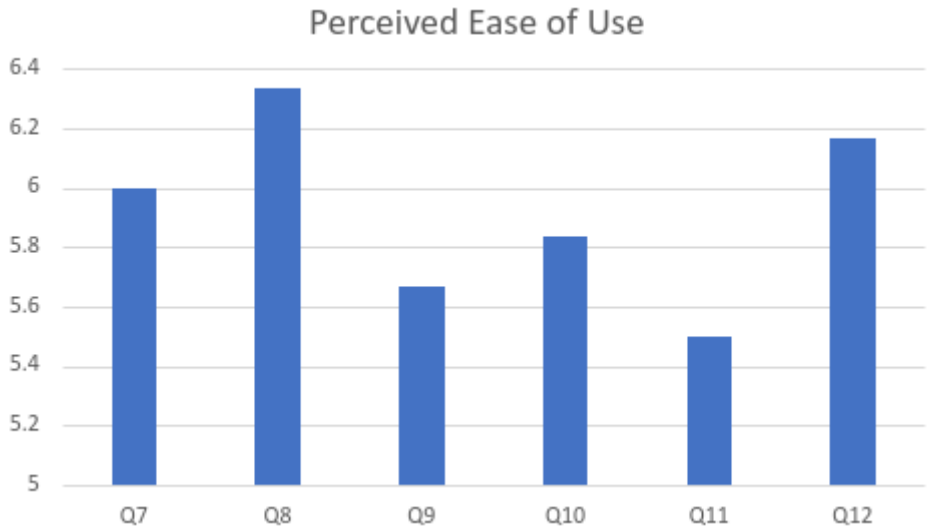


Figure 4.28: TAM perceived ease of use detailed questions results.

Regarding the qualitative data, I analyzed the observer notes and participants' answers to the open questions to identify the most important cons of the platform. Regarding the cons, P1 and P4 complained that "*[...] there is a lot of reading to grasp the meaning of each item[...]*". P6 also noted that "*[...] there is not a way to filter the cards based on documentation sources: if I need to use Microsoft Azure, I would like to filter the documentation based on Microsoft's ones [...]*". Conversely, numerous participants described the platform as easy to use and responsive (e.g., participants P1, P5, P6). P1 highlighted that it is "*[...] easy to filter results [...]*".

Suggested improvements

Our study demonstrated that the POLARIS web UI is generally perceived as effective and user-friendly. However, the study allowed us to shed light on some limitations and areas for improvement. Regardless of the platform's discovered positive attributes, some issues persist. Notably, participants felt overburdened the initial times they had to study the content of the cards to solve a task. In particular, less experienced users in TAI struggled more. Based on the insights gained from the user tests, the next iteration of platform development will incorporate the following enhancements:

- Possibility to upload a dataset and obtain automatic suggestions to implement TAI on that dataset.
- Enhance the results by adding auto-extracted tags (e.g. using a RAG component) and emphasizing (e.g. using bold text) the searched keyword.
- A filter for documentation sources.

By implementing these enhancements, I aim to address the identified limitations, providing users with an even more effective and intuitive platform for selecting the appropriate guidelines. By interviewing more users through the focus groups with different companies of different sizes, I am are sure I will still get new and useful advice insights and further enhance Polaris.

4.4 GenAI Social Impact

During the research period, I conducted some additional case studies to investigate new research fields. One of these studies aimed to investigate how ChatGPT had an impact on society, specifically regarding law comprehension.

This chapter describes the work conducted in [5].
Before explaining the work and results of this study, here some context is given.

Context

Models like ChatGPT personalize the digital version of the Delphic oracle, where people expect to find answers to their current problems by automating tasks, seeking ChatGPT's opinion on various issues, and even requesting advice. Nonetheless, it is essential to question whether we are genuinely resolving uncertainties or uncovering new ones regarding the scope, boundaries, and prospects of generative models' societal impact. Some other analysts have referred to an "AI arms race" in which companies worldwide strive to showcase the best technology, innovation prowess, and leadership in the AI market. On the other side of the debate, discussions refer to the rapid development of these models and the effectiveness of existing legal frameworks in safeguarding against unintended adverse outcomes. Amongst all these debates, unquestionably, Generative AI is currently undergoing a period of accelerated evolution. This evolution inevitably brings about a social impact akin to the ones experienced through numerous other technological

advancements that transformed our society in the past. To date, significant effects have been observed in service provision, education, and scientific analysis. However, more profound and concerning impacts also unfold in domains like democracy, inequality, security, and military technology.

Consequently, a comprehensive examination and analysis are required to understand the positive and negative social consequences, emerging trends, and areas of improvement of generative models. These studies are needed to address potential vulnerabilities and ensure the development of these technologies considers the diverse social contexts and realities in which they are deployed.

Building upon the preceding insights, this analysis adopts a comprehensive approach to explore the societal ramifications and future trajectories of Generative AI, with a specific emphasis on ChatGPT.

Study Design

To perform this review, I and some other researchers followed the protocol proposed in [285], and we completed the review process with the strategies presented in [286] for performing systematic literature reviews. The following subsections describe in detail the study design and its execution. The literature review presented in this work was carried out through the following steps:

1. **Goal and Research questions**: the goal and the correlated research questions were identified to guide the literature review;
2. **Search strategy**: defining the strategy to collect previous works published in the literature, including research databases and query strings;
3. **Eligibility criteria definition**: the criteria used to filter the collected studies have been defined;
4. **Data extraction**: defining how relevant data were extracted to help answer the research questions;
5. **Data analysis and synthesis**: defining how to organize extracted relevant data to answer the research questions.

Fig. 4.29 summarizes the review protocol.

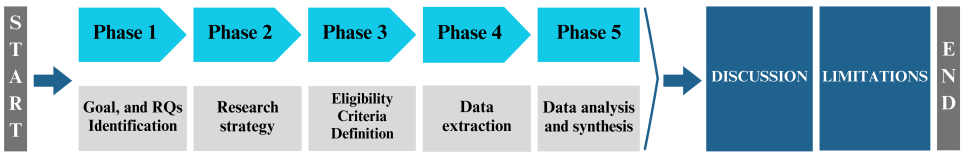


Figure 4.29: Research protocol used in the literature review

Goal and Research Question Definition

We formulated the following research questions to analyze the diverse dimensions of ChatGPT's impact.

Based on this goal, we defined the following research questions:

- **RQ1**: What are the perceived *positive and negative impacts* of ChatGPT in contemporary society?
- **RQ2**: What are the *emerging trends perceived* in ChatGPT development?
- **RQ3**: Which *areas of improvement* can be identified in the development of such technologies?

Search strategy

The research has been split into two distinct parts: *part one*, focused on grey literature, and *part two*, focused on white literature. Including both grey literature and academic contributions allowed us to conduct a deeper exploration of the impact of ChatGPT in various settings and from various perspectives.

In part one – started on 29th November 2023 – we focused on grey-literature sources, like blog posts and news articles from multiple domains – such as business, education, technology, and society – emphasizing ChatGPT. Here our goal was to feel the sentiment of the media and tech sphere and capture feelings on ChatGPT that cannot emerge from white-literature sources. Furthermore, some pilot searches on white-literature sources produced very few results, which did not allow us to derive reliable and consistent conclusions.

The string used for part one was:

> ("ChatGPT" AND "social concerns") ("ChatGPT" AND "social impact") ("ChatGPT" AND "Human rights") ("ChatGPT" AND "society*") ("ChatGPT" AND "education*") ("ChatGPT" AND "ethics")

This string was used to perform a keyword-based search on Google search engine[31]; the search was performed in private-browsing mode, after logging out from personal accounts and erasing all web cookies and history [287].

In the end, this search resulted in 1230 literature sources.

While executing part one of the research, we continued executing periodic pilot searches for white-literature sources.

In late February 2023, we noticed a notable change: the number of scientific articles addressing ChatGPT increased significantly, encompassing diverse approaches and perspectives.

[31] `https://www.google.it/`

This may be due to the fact that academic papers need more time to be reviewed by peers and published (w.r.t. blog posts and news articles).

So, for part two of the research – started on 22nd February 2023 – we decided to use Google Scholar[32] as white-literature search engine; here we searched for scientific articles of various fields – such as business, education, technology, society, healthcare. The string used for part two of the research was the following:

("Large Language Model" AND "Social impact") ("Large Language Model" AND Human Rights") ("Large Language Model" OR "ChatGPT" AND "Ethics")("Large Language Model" AND "ChatGPT" AND "Ethics") ("Large Language Model" OR "ChatGPT" AND "Social concerns") ("Large Language Model" AND "ChatGPT" AND "society*") ("Large Language Model" AND "ChatGPT" AND "education*") ("Chat GPT" AND "social concerns") ("ChatGPT" AND "social impact") ("ChatGPT" AND "Human rights") ("ChatGPT" AND "society*") ("ChatGPT" AND "education*") ("ChatGPT" AND "ethics")

In the end, this search resulted in 86 new literature sources.

All the documents obtained with this search strategy (both in part one and part two) were surveyed using a 3-stages information classification process. In the first stage, only the title and keywords of the collected articles were read. In the second stage, we analyzed the abstract of each article while in the third stage we read the complete article. All these stages were conducted separately and in blind-view way by two of the authors. In case of a disagreement, a third author manually verified and took the final decision.

All found publications were subjected to the selection criteria outlined in Sec. 4.4 to determine their relevance for inclusion in the analysis.

Eligibility Criteria Definition and Data Extraction

The selection procedure used for filtering the identified pool of 1316 papers was based on the following criteria:

- for every text the content should be mainly related to ChatGPT and LLM,
- the content should be written in English.

Then, the following criteria helped us to ensure we only included publications providing substantial information to our analysis, especially on ChatGPT, while we excluded papers with only brief mentions or tangential references:

- For *blog posts*, we required the author's name to be consistently provided, and the blogs should be specialized in the relevant subject matter.

[32] `https://scholar.google.com/`

- Regarding *news articles*, we preferred those that offered an extensive analysis. We adopted this criterion because initial news coverage of ChatGPT tended to be repetitive, often focusing on its capabilities and limitations and just providing a brief history of the model.

- In the case of *academic articles*, we sought diverse approaches to ensure a comprehensive perspective rather than lean solely on a single field, such as, for instance, the impact of ChatGPT in education.

After applying these selection criteria, we selected a total of **71** papers from our initial pool of **1316** articles.

In the Data Extraction step, we extracted all relevant data that could help answer any of the research questions. The extraction process was performed by two of the authors and conflicts were solved by a third author in a blind-view way. We used Atlas.ti[33] to tabulate and organize data. More detailed information regarding the data and how it was indexed can be found in the online appendix [11].

Data Analysis and synthesis

Part one of the search has been conducted from 29th November 2022 to 22nd February 2023. Part two has been conducted from 22nd February 2023 to 19th May 2023. Table 4.9 details the results obtained in both parts, as well as the documents selected once our selection criteria were applied.

Research phase	**Resources retrieved**	**Resources analyzed**	**Resources selected**
First part, until Feb. 22 on Google	1230	300	25
Second part, until May 19 on Google scholar	86	63	46

Table 4.9: Amount of documents collected grouped by research phase.

In order to answer RQ1, RQ2, and RQ3 we performed an analysis using Atlas.ti. Atlas.ti is a qualitative research tool that enables the systematic organization of documentary resources by using codes and creating documentary categories. Atlas.ti allowed us to visualize and intuitively present content trends within the analyzed documents. We employed this tool to analyze the selected papers, which were organized in Atlas.ti "document groups" following the categories in Table 4.10.

The categories and codes presented in Table 2 are derived from an in-depth analysis of the selected papers. The codes were primarily developed "in vivo," meaning they emerged as potential units of analysis during the reading and analysis process. For example, when repetitive references were made to the potential **positive impacts** of ChatGPT in education, we created the code "*Benefits for education*". After reviewing and analyzing the documents, the codes were organized into four categories, which helped to address RQ1, RQ2, and RQ3. The criteria for

[33] `https://atlasti.com/`

Codes for text analysis in Atlas.ti			
Positive Impact	**Negative Impact**	**Emerging trends**	**Areas for improvement**
Benefits to education	Disinformation risk	Impact on the tech/AI market	Need for appropriate regulation
Benefits to customer service	Negative impact on freedom of expression	Copyright uncertainty	GDPR compliance concerns
Benefits of responses in real-time	Bias concerns	Uncertainty over liability for production failures	Uncertainty of classification under the AI Act
...	...	...	...

Table 4.10: Excerpt of categories and codes used for analysis in Atlas.ti. Complete information can be found in Table A1 in the online Appendix [11].

categorization emerged from a thoughtful reflection on the codes and their contextual relevance. In the category of positive impacts, codes such as "*24/7 Availability*" and "*Personalized feedback*" are included since they are frequently mentioned as strengths of ChatGPT.

On the other hand, the category of **negative impacts** encompasses codes such as "*Bias concerns*", which emerged as a recurring argument when discussing the potentially detrimental effects of this model. Other codes like "*Privacy concern*" and "*Water footprint*" were identified, further emphasizing the importance of addressing these issues in the context of ChatGPT's implementation.

In the category of **emerging trends**, we captured the unforeseen consequences, which, although potentially negative, arise unexpectedly from the evolution of the model itself. Examples include "*Copyright uncertainty*" and the "*Need to clarify private sector liability*", which pose challenges that trigger transformations in particular domains such as the "*Impact on the tech/AI market*". This category also encompasses unexpected challenges to the AI Act [333] and its coverage of generative models. Another aspect within this category is "*Unintentional misinformation*", referring to instances where the chat model provides unintentionally inaccurate information, a matter currently under scrutiny by various experts. Additionally, the category encompasses codes like "*Skepticism about its actual impact*". These emerging trends shed light on the complex issues ChatGPT presents.

The final category encompasses **areas of improvement**, focusing on aspects that require further development to address all the adverse and unexpected effects of the model. For instance, the tag "*Negative outcomes mitigation*" highlights the need for more efforts to minimize adverse consequences. The categorization of codes as "*Uncertainty in data governance*" also responds to the criteria as an area of improvement rather than a negative impact. The above-mentioned decision was made considering that current legal frameworks do not adequately account for models like GPT, thus highlighting the need for specific measures to address these cases, which will likely be developed in the coming years. This category also encompasses improvement areas, such as "*Limited up-to-date information*" and "*Limited Medical terminology*", emphasizing the potential for enhancements.

After establishing the codes and categories, we computed the frequency of each code across

the 71 documents.

Table 4.11 highlights the least and most recurrent codes. Fig. A5 in the online Appendix displays the frequency of the most repeated codes.

Most recurrent	#	**Least recurrent**	#
Bias concerns	39	Unpredictability risk	1
Disinformation risk	25	Prone to injection attacks	1
Benefits for education	22	Over-regulation risk	1
Privacy concern	21	Opportunity to increase renewable energy use	1
Need for appropriate regulation	20	Need for using Renewable Energy Sources	1
Discrimination risk	18	Need for human rights safeguards	1
Unfairness in the data use in the model	18	Need for explainability and traceability	1
Benefits for customer service	17	Need for Accessibility and Affordability	1
Inaccurate answers	17	Limited Medical terminology	1
Benefits for content creation	15	Lead people into extremist positions risk	1

Table 4.11: Most and least recurrent codes; each category is associated with its number of occurrences.

In addition, Fig.4.30 displays a Sankey diagram that graphically depicts the distribution of code categories across the analyzed documents (in document groups-unit). The diagram shows that scientific papers, blog posts, conference symposiums, and other types of publications encompass all coding groups (negative and positive impacts, areas of improvement, and emerging trends). See Table A2 in online appendix [11] for a description of document categories. The articles category includes just emerging trends, areas for improvement, and negative perceptions. Most document categories, including scientific papers, columns, analyses, editorials, and articles, exhibit a negative tendency. Notably, positive perceptions are more prevalent in blog posts, conference, and symposium papers, and to a lesser degree in news articles. "*Bias concern*" and "*Disinformation risk*" are two of the most common codes contributing to negative perceptions. In contrast, "*Benefits for customer service*" and "*Multidisciplinary benefits*" are the most prevalent codes in the documents with a positive trend. Table A3 in online appendix [11] details tendencies and code frequency across all document categories.

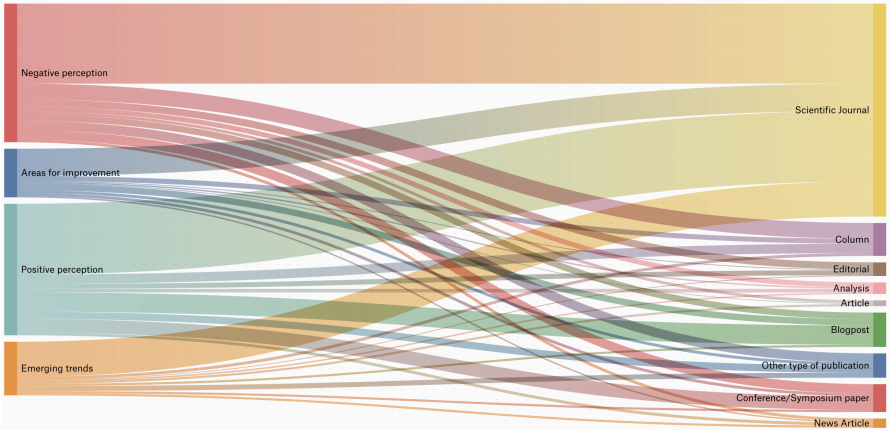


Figure 4.30: Sankey diagram illustrating the distribution of code groups across document groups

Discussion

RQ1. *What are the perceived positive and negative impacts of ChatGPT in contemporary society?*

Throughout our literature review, we identified several positive and negative impacts attributed to ChatGPT. Noteworthy benefits include: **the potential for enhancing customer service;** multiple papers emphasize the positive impact of ChatGPT in this domain [334–349]. The model is highlighted as an enabler of cross-cultural dialogue, facilitating communication between individuals from different cultural backgrounds [334]. Moreover, ChatGPT offers the advantage of **automating repetitive tasks**, freeing time for more complex and value-added activities [338, 343, 347, 350]. These benefits extend to various sectors, including business and healthcare [340]. Another key advantage is its **availability around the clock**. This 24/7 accessibility proves valuable in commercial, healthcare, and educational contexts [336, 343, 351–353]. The model's continuous availability ensures timely assistance and support, covering users' diverse needs.

ChatGPT also demonstrates significant **potential in education**, offering various advantages [335,341,344,345,348,349,352,354,355,355–361]. Despite concerns about plagiarism and academic integrity, the model's integration can enhance teaching practices in several ways. It can, for example, automates curriculum creation, enabling educators to save time and streamline the process [344, 357]. Moreover, it facilitates the development of innovative educational content, fostering an engaging learning environment [351, 354]. Additionally, the model serves as a personalized study support assistant, providing tailored guidance and assistance to individual learners [335, 341, 345, 352, 356, 358]. In this perspective, a vision emphasizes the need for controlled integration and adherence to academic guidelines to ensure responsible and ethical use of generative AI models in education [352]. By establishing appropriate regulations and ethical frameworks, the educational benefits of ChatGPT can be maximized while addressing concerns related to plagiarism and promoting an enriching learning experience.

In the medical field, the model has shown promise in research, data analysis, and telemedicine applications, **contributing to advancements in healthcare** [357]. Furthermore, diverse papers recognize its potential contribution to **addressing environmental challenges**. For instance, it may help find innovative solutions to reduce water consumption in the AI industry, highlighting its role in promoting sustainability [362]. Similarly, ChatGPT is also appreciated for its potential as an **informative and accountability tool within institutions** [335, 353, 363]. Lastly, it positively impacts journalism, where it can help with content creation, fact-checking, and generating engaging narratives [338, 344, 357].

Conversely, the most recurrent social concern is **bias**. Several papers [334–336, 336, 350, 351, 354, 358–361, 363–375] discuss the risk of deepening existing biases and how ChatGPT can include sexist and racist views due to the characteristics of the data used during its training. In this sense, there is a special concern about using these tools in various sectors, including finance [350] and other social activities, which can replicate and deepen structural

and historical inequalities. Our analysis also reveals another perceived negative impact, which is its potential to generate **false information** and facilitate the spread of disinformation [335, 336, 336–339, 346, 350, 357, 359, 360, 363, 366, 367, 369–371, 373, 376–380]. This misuse of technology directly affects rights related to access to accurate information and freedom of expression, as well as democratic stability. This phenomenon is particularly relevant as, although remarkable in artificial intelligence, the advancements and improvements in generative models have raised concerns and uncertainties regarding their **impact on democratic processes**. There is a special concern about the ease with which false information can be generated and disseminated, especially in critical contexts such as elections, referendums, political instability, war conflicts, or under dictatorial regimes. Furthermore, this potential negative impact includes the digital public sphere, where spreading **hate speech** on social networks [381] can lead to social fragmentation and bolster the manipulation of democratic institutions and social control. False information can be weaponized across various domains, from the stock market to information warfare and propaganda.

In the same vein, another relevant concern is **privacy** [334–336, 336, 340, 346, 349–351, 356, 357, 360, 363, 365, 367–370, 376, 379, 382, 383]. Different aspects contribute to this concern, including the coverage of these models under AI Act regulations, its extensive use of user data (particularly minors), the potential for surveillance applications, the data vulnerability to cyber attacks targeting ChatGPT, and the privacy implications when integrating the model into various domains such as education and the military. Privacy is a growing concern as illustrated by the case of the Italian state ban. In this case, the Italian authorities asked OpenAI to "expand its privacy policy for users and made it also accessible from the sign-up page prior to registration with the service" [384] in order to operate in Italy. This case highlights the urgency for ensuring responsible and transparent use of the technology. Another important negative impact is **the risk of job loss**, as automation and AI capabilities advance [338, 340, 341, 349, 350, 355, 364, 367, 375, 377, 378, 385, 386].

Furthermore, other papers show apprehension about **over-regulation** [376], highlighting the needed balance between ensuring ethical and responsible use of AI technologies while avoiding stifling innovation. Additionally, the model's own cybersecurity is listed as a negative impact [343, 346, 350, 357, 365, 376, 377]. The comprehensive list of negative perceptions on ChatGPT impacts can be found in Table A1 in online appendix [11], providing further insights into the concerns found in the literature.

RQ2. *What are the emerging trends perceived in ChatGPT development?*

An essential part of our review shows emerging trends [335–337, 340–343, 346, 348–350, 352, 355, 355, 357–362, 366, 368, 370, 371, 375, 379–383, 385, 387–392]. Within this category, we found **uncertainty surrounding copyright**, which is a prominent trend, raising doubts about the possibility of profiting from chat-generated content and whether such content is subject to copyright protection. Resolving these issues requires legislative intervention, leading to a particular and complex debate among authorities regarding the legal implications of AI-

generated content [337]. There is also a growing demand for the **enhancement of regulatory frameworks** to safeguard original content, considering that models like ChatGPT have the potential to negatively impact the work of scientists, writers, researchers, and artists [335, 346, 349, 360, 375, 389]. Notably, analyses such as by Khowaja et al. [346] raise crucial questions regarding ownership rights over the data used to train the model and the ownership of the model itself. As mentioned earlier, there is a dominant call for establishing **ethical use guidelines**, both at a general societal level [386] and specifically within educational and research institutions [348, 349, 355, 356, 358–360, 371, 375, 383, 386]. These guidelines will be pivotal in ensuring responsible adoption models such as ChatGPT.

Another emerging trend is developing **transparency mechanisms** [335,346,350,351,355,357, 360, 366, 375]. Transparency is considered a vital strategy to address resistance toward adopting these models in various contexts [357] and to mitigate potential AI stigmatization. However, it is also acknowledged that transparency poses challenges that need to be overcome [346].

Another crucial issue to highlight is the **accountability for potentially harmful uses** of such technology [335, 336, 340, 346, 370, 375, 376, 382], which involves both the end-users and the companies responsible for its development [335, 336, 340, 346, 362, 370, 375, 376, 381, 382]. This concern extends to applications in sensitive domains like the military [340]. Furthermore, other several notable trends emerge, including the need for **timely and appropriate regulation** [348, 386], the existence of **political bias** (29, 40, 42, 62), **transformations within the AI market** [340, 342, 361, 366, 368, 371, 373], and the **integration of renewable technologies and environmental awareness** within this field [334, 365].

RQ3. *Which areas of improvement can be identified in the development of such technologies?*

The findings concerning areas for improvement within the field of generative AI present a diverse range of perspectives [335–337, 340, 342, 343, 343, 345–350, 352, 357, 359, 360, 363–366, 368, 371, 375–379, 381, 386, 389, 393–395]. One prominent area is the examination of regulations [335, 337, 340, 342, 343, 348, 355, 359, 360, 362, 364, 366, 371, 376, 378, 381, 386, 389], particularly regarding whether the risk-based approach outlined in the **AI Act** effectively covers generative models [376, 378]. It is suggested that comprehensive guidelines should encompass the entire spectrum, from its application to the AI Research and Development (R&D) [362]. Furthermore, there is a growing advocacy for **a people-centered vision** [335, 342, 378] that emphasizes the importance of human rights and ethical considerations in designing and implementing generative AI systems. Specifically, Solaiman et al. [359] examine the need to "build frameworks for navigating trade-offs" and develop decision-making frameworks that account for the complexities and potential trade-offs associated with generative AI. Similarly, Li [343] highlights the necessity to transform the European regulatory paradigm to effectively address the challenges posed by LLMs.

Another area of opportunity lies in addressing **technical limitations** within generative AI models [336, 343, 345, 347, 349, 352, 357, 360, 371, 374, 385, 395, 396]. For instance, a significant challenge is the presence of **fictional references** [360] within the generated text, which hampers

its reliability. Additionally, **words repetition** further affects the produced text's overall quality [378]. Moreover, the phenomenon of inaccurate information, named "*hallucinations*" [371, 378], and the lack of context understanding [336], are also identified as areas requiring attention and improvement. Furthermore, the literature highlights the need for enhanced **risk mitigation mechanisms** [343, 364, 366, 368, 377, 379, 393]. This entails **refining processes for filtering potentially harmful responses** [364], **improving the quality and reliability of the data used to train the models** [343], and **incorporating ethical guidelines** for its development [377], among other measures [368].

Data governance is another significant area that requires attention [335, 340, 350, 360]; this crucial task involves safeguarding sensitive information against security breaches, unauthorized access, and information theft [335, 350]. In the same vein, **establishing clear guidelines regarding the scope and limitations of information exchange with third parties** is also paramount [360].

Other areas of opportunity include **the need for up-to-date data** [345, 347, 352, 394] to ensure the accuracy and relevance of generative AI models. However, this requirement presents a trade-off between incorporating new data to improve performance or addressing data governance issues first.

The literature review also highlights several other areas of opportunity for improvement; these include promoting **end-user responsibilities** [369], advocating for **timely regulation** [360, 369], and **raising awareness of environmental impact** [360, 361], among other considerations.

4.5 Leading Companies TAI Practices comparison

I conducted another additional case study to investigate new research fields. This second study aimed to analyze and compare the strategies and actions taken by four leading companies — OpenAI, Meta AI Research, Google AI, and Microsoft AI — regarding how they implement Trustworthy AI.

This chapter describes a summary of the work conducted in [8].
Before explaining the work and results of this study, here some context is given.

Context

Generative Artificial Intelligence (AI) has made significant progress in recent years, providing advantages across diverse sectors, including the creative industry and medicine [397, 398]. Nevertheless, there are domains and contexts in which Generative AI can impact human rights, including but not limited to privacy, non-discrimination, freedom of opinion and expression, freedom of peaceful assembly and various fundamental freedoms. Within this framework, Foundation Models are a rapidly expanding sector. For instance, after its launch, ChatGPT — developed by OpenAI — gathered a user base of over 100 million within less than two

months [399]. These developments underscore the pressing need for the progression of *responsible* and *rights-committed* Generative AI.

In this scenario, the private sector — particularly companies specializing in AI development — assumes a pivotal role. In this respect, the Office of the High Commissioner for Human Rights (OHCHR) recognizes two distinct approaches to regulating Artificial Intelligence [400]. The first approach is a risk-based strategy that assigns substantial accountability to the private sector by identifying and addressing risks to attain desired results. The second incorporates human rights principles into every AI development and implementation stage. This approach includes human rights principles throughout the data collection and selection process and the design, development, deployment, and use of the resulting models, tools, and services. As per OHCHR, immediate measures are required today to prevent, tackle, and mitigate the potential adverse effects of Generative AI [400].

The human rights issue in the AI industry is a topic of ongoing and contentious discussion. Various organizations, including the United Nations [401], International Amnesty [402], European Center for Not-for-Profit Law Stichting [403], Access Now [404], AI Now Institute [405], and others, actively seek to collaborate with the industry to advance Responsible AI.

This work aims to contribute to these efforts by attempting to fill the gap between high-level principles and low-level concrete practices that can be deployed by the AI industry to address the challenges and concerns of one particular area of Responsible AI, the one connected to human rights protection. The human rights doctrine, on one side, can contribute to closing the gap between high-level AI ethics principles and low-level concrete practice. On the other side, it can support the definition of AI ethics principles, as human rights can support value alignment for AI systems across a range of different national and social contexts [406]. The human rights vision can help close the gap between research and practice and between scientists and civil society, providing a common vocabulary and a shared understanding [406].

The outcomes of our research yield the following research contributions:

- Through a **systematic review**, I and some other researchers have highlighted the specific measures taken by a set of big AI players (i.e. OpenAI, Meta AI Research, Google AI, and Microsoft AI) in addressing five different dimensions: *Bias*, *Misinformation*, *Hate speech*, *Cybersecurity*, and *Privacy*. These actions directly influence the protection of fundamental human rights, encompassing but not limited to the right to non-discrimination, health, security, access to information, free expression, and privacy, among others. Additionally, our assessment has identified both **strengths** and **weaknesses** in the company's measures and strategies.

- By conducting a **thematic analysis** of the most important publicly available documents related to each parameter in each selected organization, we have identified recurring themes that revolve around ethics, the need for diversity and representation, and the importance placed on collaboration and research.

- In light of these findings, we offer a set of **actionable best practices** that these and other companies can embrace to enhance their performance on these parameters. This, in turn, will strengthen their approach to Responsible and Trustworthy AI, solidifying their commitment to upholding human rights.

Building upon the research conducted in the study addressing the maturity of Responsible AI in [4] and the examination of challenges and gaps within the AI industry [407], our work makes a valuable contribution by proposing a series of best practices to be implemented within the AI industry. It is essential to emphasize that these recommendations draw inspiration from existing measures while also presenting a potential remedy for identified weaknesses. Collectively, these practices intend to uphold human rights within the burgeoning AI industry.

Methodology

This qualitative study conducts a descriptive and comparative analysis of the measures implemented by four leading AI companies regarding *bias* reduction, *privacy* protection, *cybersecurity*, fighting *hate speech*, and addressing *misinformation*. We conducted a systematic (grey) literature [408] review of all the publicly available documents of these companies, followed by a comparative analysis.

Research Questions

Our research questions are as follows:

- **RQ1.** What Responsible AI practices, in terms of **risk mitigation and human rights protection strategies/actions**, do leading AI companies like OpenAI, Google, Microsoft, and Meta employ when developing AI models, and in particular Generative AI models?
- **RQ2.** What are companies' **strengths**, and **weaknesses** in risk mitigation and human rights protection actions?
- **RQ3.** Which **best practices** can be adopted to enhance risk mitigation and human rights safeguard within the AI industry, with particular reference to Generative AI development?

Study design

To conduct our study, we implemented the following steps:

1. **Companies and parameters selection**: This phase involves identifying and selecting the specific companies to be analyzed and determining the evaluation's parameters or criteria.
2. **Establishing a search strategy**: A systematic strategy is defined to find relevant information in the published document sources.

3. **Adopting eligibility criteria for data extraction**: This is done in order to ensure that only meaningful and reliable sources are included.

4. **Analysing and synthesizing selected data**: The obtained data is evaluated, organized, and synthesized to make a meaningful analysis.

5. **Performing thematic analysis**: Using Atlas.ti we identified recurring themes within the synthesized data, providing a deeper understanding of the overarching trends and characteristics.

The following sections provide a detailed description of each step.

Companies and Parameters Selection

In order to analyze how some of the Responsible AI issues are addressed by the industry, we conducted a comprehensive analysis of **four leading companies** operating within the AI industry: OpenAI, Meta AI Research, Google AI, and Microsoft AI. These companies were selected based on their substantial impact on the AI market and research, and their relevance across various domains [409].

Each of the four aforementioned companies exhibits a wide range of approaches and regulations for Responsible AI development, each with its own particular mission and vision.

OpenAI is a notable corporation renowned for its innovative work in Artificial Intelligence and great emphasis on research. OpenAI's mission is to create safe Artificial General Intelligence for humanity's benefit. ChatGPT and Dall-E are two of its products, and its research spans across the text, audio, and image domains [410].

Meta AI Research is a Meta Platforms, Inc. division. Meta's Fundamental AI Research Team (FAIR) is devoted to various AI-related topics. Meta's vision is to disseminate its research within the AI community and encourage collaborative efforts to develop Responsible AI [411].

Google AI has emerged as a prominent and influential participant in AI, making significant contributions. Google AI is committed to performing impactful AI research to enhance societal well-being. It actively participates in the academic community by sharing its research outcomes through open-source projects and facilitating global collaborations [412].

Microsoft AI has the objective of facilitating the empowerment of people and organizations by democratizing AI [413]. Azure, the Microsoft cloud computing platform, plays a vital role within their AI ecosystem by facilitating the deployment and management of AI applications and solutions [413].

The selection of the parameters for the analysis is based on the insights obtained from prior scholarly research into the perceived impacts (positive and negative) of Generative Artificial Intelligence [398] and the human rights challenges in AI [407]. Thus, based on the concerns identified in [398] and [407], we have compiled a preliminary inventory of rights that could be undermined by the misuse of Generative AI or by insufficient protective measures. A

comprehensive list can be found in the online appendix [414]. In this initial research, we have focused on five specific issues that will serve as parameters. In the future, we plan to broaden our analysis to include other criteria such as algorithmic transparency, liability for damages caused, etc.. With the above in mind, the five parameters chosen are as follows:

1. **Bias**. Following the definition given in [415], we define bias as "*an inclination of prejudice towards or against a person, object, or position*". The potential for undesirable biases in AI systems to exacerbate existing social inequities — or even generate new ones — has recently received considerable attention across a range of academic disciplines, from AI to SE to public policy, law, and ethics [416], [417], [418]. Bias reduction is related to the rights of Equality and Non-Discrimination.

2. **Misinformation**. Is fake or inaccurate information, while disinformation is deliberately created to deceive or manipulate [419]. Fighting dis/misinformation is linked to protecting the right to information, freedom of expression, and participation in public affairs. Depending on the misinformation content or intention, it may also affect rights such as health, non-discrimination, life, and personal security, among others. **Note:** In this study, we employ the term misinformation because, regardless of whether it is part of an orchestrated operation or not, false information has a variety of detrimental effects on society — some of the companies' strategies analysed focus on disinformation or misinformation.

3. **Hate Speech**. It is defined as a type of offensive language that uses stereotypes to express a hateful ideology [420]. It is further described as any communication that disparages a person or group based on a characteristic such as race, color, ethnicity, gender, sexual orientation, national origin, religion, etc. [421]. Fighting hate speech is related to protecting rights such as the right to equality and non-discrimination, freedom of thought and expression, security of person, public order and safety.

4. **Privacy**. There is a large discussion on its definition; while some consider it "*the state in which a person is not observed or disturbed, or to be free from public attention*" [422] others hold that "*is not simply an absence of information about us in the minds of others, rather it is the control we have over information about ourselves*" [423]. For the purposes of this research, we refer to privacy as the right to have one's data processed in a way that results compliant with the seven protection and accountability principles outlined in GDPR Article 5.1-2 [305]: *Lawfulness, fairness and transparency*, *Purpose limitation*, *Data minimization*, *Accuracy*, *Storage limitation*, *Integrity and confidentiality*, and *Accountability*. Privacy is a human right in itself and is related to others, such as the protection from Unlawful Interference with Privacy and Respect for Private and Family Life.

5. **Cybersecurity**. According to the European Union Agency for Cybersecurity (ENISA), "*shall refer to security of cyberspace, where cyberspace itself refers to the set of links and*

relationships between objects that are accessible through a generalised telecommunications network, and to the set of objects themselves where they present interfaces allowing their remote control, remote access to data, or their participation in control actions within that Cyberspace" [424]. Since data is mentioned in this definition too, and Cybersecurity itself is considered an implementable measure into GDPR [305], it is often difficult to mark a separation line between Privacy and Cybersecurity. That is why we considered also some specific data protection nuances. Cybersecurity is related to rights such as the Security of Person and Privacy.

The rights listed above are enshrined in the *Universal Declaration of Human Rights* (UDHR), the *International Covenant on Civil and Political Rights* (ICCPR), and the *European Convention on Human Rights* (ECHR). We have included these rights as they are directly relevant to the risks identified in the literature. However, it is essential to remember that human rights are indivisible, interrelated, and interdependent [425]. Therefore, additional rights may be compromised if, for instance, biases in Generative AI persist.

Discussion of Results

RQ1. What Responsible AI practices, in terms of *risk mitigation and human rights protection strategies/actions*, do leading AI companies like OpenAI, Google, Microsoft, and Meta employ when developing AI models, and in particular Generative AI models? The measures and efforts related to human rights are strongly influenced by the diversity of services and industry experience in AI. For instance, companies like Google AI and Meta AI Research stand out in combating misinformation due to the array of products they offer, such as social media platforms and free news aggregator services. This grants them influence but also entails a significant responsibility in addressing issues like misinformation and hate speech. On the other hand, OpenAI, by providing Generative-AI-based services and having a narrower range of products, can contribute to protecting human rights, such as privacy, through the research of novel techniques for privacy preservation. Meanwhile, Microsoft AI's experience and services enable it to bolster cybersecurity techniques, particularly valuable in contexts like elections to protect the systems from hacking.

Furthermore, OpenAI and Microsoft AI, for instance, exhibit a proactive involvement in pioneering research, forging partnerships with leading institutes and research centres to push the boundaries of AI capabilities. In contrast, Google AI and Meta AI Research place a pronounced emphasis on fostering collaboration with governmental bodies.

RQ2. What are companies' *strengths*, and *weaknesses* in risk mitigation and human rights protection actions? Strengths include in certain cases, accessibility to information certifications compliance, access to findings from various research endeavors, as well as databases, and the ability to exercise control over personal data.

Additionally, there is a growing awareness of the need for interdisciplinary collaboration, which will undoubtedly contribute to fortifying protection mechanisms. Another strength lies in their active and continuous engagement in research and development, a growing trend towards investigating the social impact of technologies, as well as the creation of specialized teams in collaboration with experts in the field of human rights.

However, there are notable weaknesses to address. One significant area is the limited cultural, social, and linguistic diversity. This deficit results in a potential bias towards Western perspectives in research lines and tools for addressing bias, privacy, and cybersecurity, combating hate speech, and countering misinformation. Furthermore, it is vital to broaden and vary the collaboration with organizations, institutions, and governments across various regions of the globe. On the other hand, it is also necessary for data control tools to be accessible to all population sectors, especially individuals with disabilities.

In a broader sense, there is a lack of engagement with non-governmental organizations, civil associations, and institutes dedicated to AI ethics and human rights protection.

RQ3. Which best practices can be adopted to enhance risk mitigation and human rights safeguard within the AI industry, with particular reference to Generative AI development?

Based on the findings of this research, it is recommended:

1. Actively engage with non-governmental organizations and AI ethics and human rights research centers from diverse backgrounds to enhance the approach to bias mitigation.

2. The research team, research lines, and tools on bias should encompass geographic, linguistic, social, and cultural diversity.

3. Collaborate with academia and specialized centres to deepen understanding of the scope and consequences of misinformation, involving experts from diverse regions.

4. Expand collaborative networks for election security and false content identification across several locations, especially in unstable contexts and during critical periods.

5. Generate reports outlining efforts in the field of combating misinformation.

6. Promote transparency by publishing reports on content removal requests.

7. Promote media literacy to raise awareness among the audience about the capabilities of Generative AI and their potentially harmful effects in terms of misinformation.

8. Actively conduct reviews of Generative AI to assess the likelihood of hate speech generation.

9. Collaborate with specialized institutions and research centres in ethics and human rights to comprehensively capture the complex dimensions of hate speech.

10. Share research findings with the AI community, with particular support for startups and less experienced companies on hate speech.

11. Collaborate with organizations and governments worldwide to counteract hate speech.

12. Promote privacy awareness among the general public by disseminating educational materials on privacy.

13. Foster ongoing collaboration with diverse academia, research centres, organizations, and pertinent government entities on privacy protection.

14. Ensure transparent communication regarding interactions with Generative AI models.

15. Provide users with accessible and straightforward information on applicable privacy policies, along with mechanisms for exercising control over their data.

16. Prioritize accessibility of data control centres with a particular focus on ensuring inclusivity for individuals with disabilities.

17. Sustain partnerships with research institutions, regulatory bodies, and emerging startups and continue sharing lessons learned, findings, and datasets to fortify the cybersecurity of the AI ecosystem.

18. Whenever feasible, proactively disseminate information regarding cybersecurity compliance certifications.

Chapter 5

Conclusions and Future Works

5.1 Conclusions and Future Works

In this thesis work, I achieved two macro-results:

- enhanced and enriched POSD framework for conventional software systems
- Formulated and validated POLARIS framework for AI-based software systems

Regarding the first macro-result, I provided POSD with a new web UI, experimented with it on industrial case studies, added knowledge from GDPR and ISO 9241-210 phases to its knowledge base (so realizing the *MATERIALIST* framework) and validated the need for Sonarqube and Fortify as SAST.

Anyway, a lot of work can still be done. For example:

- Automate some web UI tasks, in order to make the users spend less clicks
- Realize a web UI also for MATERIALIST framework

Regarding the second macro-result, I came up with POLARIS, a new framework designed to fill the gaps highlighted in the state of the practice by providin AI practitioners with actionable guidelines specific to each phase of the SDLC.

POLARIS has four pillars (or *components*), which are Explainability, Fairness, Security, and Privacy. These principles have been chosen as they are the most recurrent TAI principles found in the current literature [15]. Each component provides practical guidelines and tools to support different types of stakeholders throughout the SDLC.

Its added value is that it provides knowledge already freely available online, but in an organized and systematized way.

One may argue that the current literature already has similar TAI frameworks. Anyway, as Tab. 5.1 shows, I crafted POLARIS starting from what the competitors currently miss, so that POLARIS is more flexible than its competitors on key aspects: it is able to cover all four

Table 5.1: Comparison between POLARIS and the top 5 most comprehensive other competitors already published in the literature (from Section 4.1).

Framework	**Covers all Principles**	**Supported with a tool**	**For different kinds of stakeholders**	**Covers all SDLC phases**
POLARIS	X	X	X	X
PLOT4ai	X	X		X
IBM Trustworthy AI	X	X		
ENISA Securing Machine Learning Algorithms	X		X	
NIST AI Risk Management Framework	X			
NASA Framework for the Ethical Use of Artificial Intelligence (AI)				

TAI principles previously explained in Section 4.1, is it provided with a tool that supports its navigation (Section 4.3.5), can provide useful insights for both technical and non-technical stakeholders and offer supports to all SDLC phases.

However, this still does not make POLARIS the perfect framework; additional research must be done in the current literature, as new potentially better TAI frameworks emerge daily.

I applied POLARIS to an industrial case study and collected feedback through the thinking-aloud method. This allowed us to identify and apply several improvements.

POLARIS is a preliminary attempt to organize and make knowledge on TAI principles easily accessible and available to different kinds of stakeholders. It is a pioneering prototype whose goal is to make AI professionals, policymakers, and stakeholders able to navigate the ethical dimensions of TAI with confidence, ensuring that the vast potential of AI is harnessed responsibly for the benefit of society.

Even if I applied it to real-world contexts and validated its UI with a first round of user interviews, several improvements are still required. Some of them are:

- The monitoring SDLC phase is the one with the least concrete guidelines, probably because the advent of the cloud has, in some way, delegated the monitoring burden to third parties. Anyway, since the IT user had to deploy an AI-enabled system on an on-premise infrastructure, he pointed us to this lack of the framework. For this reason, it is required to conduct further research in the current literature to collect as many concrete monitoring guidelines as possible; this way, POLARIS would become more effective in the monitoring phase;

- Most knowledge sources composing POLARIS were published before the recent great improvements in Large Language Models (LLMs). As a consequence, POLARIS at the moment lacks guidelines and tools specific to LLMs. For this reason, it is required to

research, integrate and experiment with both guidelines and automatic tools to specifically address trustworthiness in LLMs;

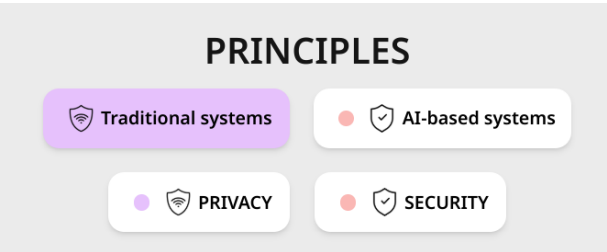
- Make the filtering by natural language (i.e. the search bar) more intelligent by adding auto-extracted tags to each suggestion and adding a more efficient filtering algorithm (e.g. a RAG, *Retrieval-Augmented Generation*);
- Emphasize more (e.g. using bold text) the searched keyword in each shown guideline.

By further testing POLARIS, I am sure I will identify new gaps that, once filled, will refine the framework.

Another important research stream regards the possibility of merging POSD and POLARIS in a unique framework, able to push the development of security- and privacy-oriented traditional and AI-enabled systems. Thanks to the modularity and extensibility of POLARIS, it could be very easy to integrate the entire PKB into POLARIS and its interface; to implement a simple filtering system similar to the current one, I could experiment with adopt two different approaches:

- Create a new page, similar to the current one named "Tools", with a different set of filters;
- Add a new set of filters in the current page named "Tools", specifically adapted for traditional software systems; an example is shown in Fig. 5.1.

Figure 5.1: Example of a new set of filters useful to integrate PKB into POSD.



By conducting various experiments on real-world projects, I will be able to discover which approach is better for the users.

Bibliography

[1] V. Barletta, G. Desolda, D. Gigante, R. Lanzilotti, M. Saltarella, From gdpr to privacy design patterns: The materialist framework, in: Proceedings of the 19th International Conference on Security and Cryptography-SECRYPT, 2022, pp. 642–648.

[2] M. T. Baldassarre, V. S. Barletta, G. Dimauro, D. Gigante, A. Pagano, A. Piccinno, Supporting secure agile development: the vis-prise tool, in: Proceedings of the 2022 International Conference on Advanced Visual Interfaces, 2022, pp. 1–3.

[3] D. Gigante, F. Pecorelli, V. S. Barletta, A. Janes, V. Lenarduzzi, D. Taibi, M. T. Baldassarre, Resolving security issues via quality-oriented refactoring: A user study, in: 2023 ACM/IEEE International Conference on Technical Debt (TechDebt), 2023, pp. 82–91. doi:10.1109/TechDebt59074.2023.00016.

[4] V. S. Barletta, D. Caivano, D. Gigante, A. Ragone, A rapid review of responsible ai frameworks: How to guide the development of ethical ai, in: Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering, EASE '23, Association for Computing Machinery, New York, NY, USA, 2023, p. 358–367. doi:10.1145/3593434.3593478.
URL https://doi.org/10.1145/3593434.3593478

[5] M. T. Baldassarre, D. Caivano, B. Fernandez Nieto, D. Gigante, A. Ragone, The social impact of generative ai: An analysis on chatgpt, in: Proceedings of the 2023 ACM Conference on Information Technology for Social Good, GoodIT '23, Association for Computing Machinery, New York, NY, USA, 2023, p. 363–373. doi:10.1145/3582515.3609555.
URL https://doi.org/10.1145/3582515.3609555

[6] M. T. Baldassarre, D. Gigante, M. Kalinowski, A. Ragone, Polaris: A framework to guide the development of trustworthy ai systems, in: Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering - Software Engineering for AI, CAIN '24, Association for Computing Machinery, New York, NY, USA, 2024, p. 200–210.

doi:10.1145/3644815.3644947.
URL https://doi.org/10.1145/3644815.3644947

[7] M. T. Baldassarre, D. Gigante, M. Kalinowski, A. Ragone, S. Tibidò, Trustworthy ai in practice: an analysis of practitioners' needs and challenges, in: Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering, EASE '24, Association for Computing Machinery, New York, NY, USA, 2024, p. 293–302. doi:10.1145/3661167.3661214.
URL https://doi.org/10.1145/3661167.3661214

[8] B. M. Teresa, C. Danilo, F. N. Berenice, G. Domenico, R. Azzurra, Fostering human rights in responsible ai: A systematic review for best practices in industry, IEEE Transactions on Artificial Intelligence (2024) 1–15doi:10.1109/TAI.2024.3394389.

[9] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks (2013). doi:10.48550/ARXIV.1312.6199.
URL https://arxiv.org/abs/1312.6199

[10] M. T. Baldassarre, D. Gigante, M. Kalinowski, A. Ragone, Polaris appendix, https://figshare.com/s/16bc31211b11c3155f81 (2023).

[11] M. T. Baldassarre, D. Caivano, B. Fernàndez Nieto, D. Gigante, A. Ragone, https://figshare.com/s/77c3a667671472f8eccc (2023).
URL https://figshare.com/s/77c3a667671472f8eccc

[12] N. F., A. A., G. S., M. P., S. S., Project Management: Driving Complexity. PMI Italian Academic Workshop, 2018. doi:10.13133/9788893770866.

[13] J. Cristóbal, L. Carral, E. Díaz, J. Fraguela, G. Iglesias, Complexity and project management: A general overview, Complexity 2018 (2018) 1–10. doi:10.1155/2018/4891286.

[14] M. Ferreiraa, A. Tereso, P. Ribeiro, G. Fernandes, I. Loureiro, Project management practices in private portuguese organizations, Procedia Technology 9 (2013) 608–617. doi:10.1016/j.protcy.2013.12.067.

[15] A. Jobin, M. Ienca, E. Vayena, The global landscape of ai ethics guidelines, Nature Machine Intelligence 1 (9) (2019) 389–399. doi:10.1038/s42256-019-0088-2.
URL https://doi.org/10.1038/s42256-019-0088-2

[16] A Guide to the Project Management Body of Knowledge, Sixth Edition, 2017.

[17] R. J. Ellison, Security and project management, 2006.

[18] C. Decker, R. Wattenhofer, Bitcoin transaction malleability and MtGox, in: Computer Security - ESORICS 2014, Springer International Publishing, 2014, pp. 313–326. doi:

10.1007/978-3-319-11212-1_18.
URL https://doi.org/10.1007%2F978-3-319-11212-1_18

[19] K. Ahmad, M. Maabreh, M. Ghaly, K. Khan, J. Qadir, A. Al-Fuqaha, Developing future human-centered smart cities: Critical analysis of smart city security, data management, and ethical challenges, Computer Science Review 43, cited by: 1; All Open Access, Bronze Open Access (2022). doi:10.1016/j.cosrev.2021.100452.
URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85121968014&doi=10.1016%2fj.cosrev.2021.100452&partnerID=40&md5=203c708b33d037b72921c18a669803cd

[20] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, D. Song, Robust physical-world attacks on deep learning models (2017). doi:10.48550/ARXIV.1707.08945.
URL https://arxiv.org/abs/1707.08945

[21] N. Carlini, D. Wagner, Audio adversarial examples: Targeted attacks on speech-to-text, in: 2018 IEEE Security and Privacy Workshops (SPW), 2018, pp. 1–7. doi:10.1109/SPW.2018.00009.

[22] E. Ackerman, Three small stickers in intersection can cause tesla autopilot to swerve into wrong lane, IEEE Spectrum (2019).

[23] S. G. Finlayson, J. D. Bowers, J. Ito, J. L. Zittrain, A. L. Beam, I. S. Kohane, Adversarial attacks on medical machine learning, Science 363 (6433) (2019) 1287–1289. arXiv:https://www.science.org/doi/pdf/10.1126/science.aaw4399, doi:10.1126/science.aaw4399.
URL https://www.science.org/doi/abs/10.1126/science.aaw4399

[24] A. A. Ari, O. K. Ngangmo, C. Titouna, O. Thiare, Kolyang, A. Mohamadou, A. Guéroui, Enabling privacy and security in cloud of things: architecture, applications, security & privacy challenges, Applied Computing and Informatics (2020).

[25] T. J. Ansari, D. Pandey, Risks, security, and privacy for hiv/aids data: Big data perspective, 2018.

[26] S. A. Spector, The Evolution of Blake's Myth (1st ed.), 2020.

[27] A. Becla, Information society and knowledge-based economy - development level and the main barriers - some remarks, Economics & Sociology 5 (2012) 125–132. doi:10.14254/2071-789X.2012/5-1/8.

[28] D. J. Solove, Conceptualizing privacy, Calif. L. Rev. 90 (2002) 1087.

[29] D. J. Solove, Understanding privacy (2008).

[30] T. M. J., The Rights of Publicity and Privacy, Thomson/West, 2006.

[31] Ico: Conducting privacy impact assessments code of practice.
URL https://uk.practicallaw.thomsonreuters.com/9-606-2025

[32] Y. K. Al-Otaibi, F. Federico, The impact of health information technology on patient safety, Saudi Medical Journal 38 (2017) 1173 – 1180.

[33] M. Zulkernine, S. Ahamed, Software security engineering: Toward unifying software engineering and security engineering, Enterprise Information Systems Assurance and System Security: Managerial and Technical Issues (2006) 215–233doi:10.4018/978-1-59140-911-3.ch014.

[34] N. Mead, T. Stehney, Security quality requirements engineering (square) methodology, ACM SIGSOFT Software Engineering Notes 30 (2005) 1–7. doi:10.1145/1082983.1083214.

[35] P. X. Mai, A. Goknil, L. K. Shar, F. Pastore, L. C. Briand, S. Shaame, Modeling security and privacy requirements: a use case-driven approach, Information and Software Technology 100 (2018) 165–182. doi:https://doi.org/10.1016/j.infsof.2018.04.007.
URL https://www.sciencedirect.com/science/article/pii/S0950584918300703

[36] E. Paja, F. Dalpiaz, P. Giorgini, Modelling and reasoning about security requirements in socio-technical systems, Data & Knowledge Engineering 98 (2015) 123–143, research on conceptual modeling. doi:https://doi.org/10.1016/j.datak.2015.07.007.
URL https://www.sciencedirect.com/science/article/pii/S0169023X1500052X

[37] M. Baldassarre, V. Barletta, D. Caivano, M. Scalera, Integrating security and privacy in software development, Software Quality Journal 28 (09 2020). doi:10.1007/s11219-020-09501-6.

[38] J. B. Earp, F. C. Payton, Dirty laundry: privacy issues for it professionals, IT Professional 2 (2) (2000) 51–54. doi:10.1109/6294.839371.

[39] European Commission, Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance) (2016).
URL https://eur-lex.europa.eu/eli/reg/2016/679/oj

[40] L. Moses, Recurring dilemmas: The law's race to keep up with technological change, University of Illinois Law, Technology & Policy 2007 (04 2007). doi:10.2139/ssrn.979861.

[41] G. Danezis, J. Domingo-Ferrer, M. Hansen, J. Hoepman, D. L. Métayer, R. Tirtea, S. Schiffner, Privacy and data protection by design - from policy to engineering, CoRR abs/1501.03726 (2015).
URL http://arxiv.org/abs/1501.03726

[42] Cis critical security controls version 8 (2023).
URL https://www.cisecurity.org/controls/v8

[43] COBIT 5: A Business Framework for the Governance and Management of Enterprise IT, COBIT® 5, ISACA, 2012.
URL https://books.google.it/books?id=1iLKVlOIg9EC

[44] I. I. O. for Standardization, ISO/IEC 27001:2022, ISO (International Organization for Standardization, CP 401 - 1214 Vernier, Geneva, Switzerland, 2022.
URL https://www.iso.org/standard/27001

[45] M. Barrett, Framework for improving critical infrastructure cybersecurity version 1.1 (2018-04-16 2018). doi:https://doi.org/10.6028/NIST.CSWP.04162018.

[46] K. Dempsey, G. Witte, D. Rike, Summary of nist sp 800-53, revision 4: Security and privacy controls for federal information systems and organizations (2014-02-19 2014). doi:https://doi.org/10.6028/NIST.CSWP.02192014.

[47] R. Ross, Protecting controlled unclassified information in nonfederal systems and organizations (2023). doi:https://doi.org/10.6028/NIST.SP.800-171r3.fpd.

[48] M. T. Baldassarre, D. Caivano, G. Visaggio, Empirical studies for innovation dissemination: Ten years of experience, in: Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering, EASE '13, Association for Computing Machinery, New York, NY, USA, 2013, p. 144–152. doi:10.1145/2460999.2461020.
URL https://doi.org/10.1145/2460999.2461020

[49] P. Ardimento, D. Caivano, M. Cimitile, G. Visaggio, Empirical investigation of the efficacy and efficiency of tools for transferring software engineering knowledge, Journal of Information & Knowledge Management (JIKM) 07 (2008) 197–207. doi:10.1142/S0219649208002081.

[50] Protecting consumer privacy in an era of rapid change: Recommendations for businesses and policymakers.
URL https://www.ftc.gov/reports/protecting-consumer-privacy-era-rapid-change-recommendations-businesses-policymakers

[51] P. de Hert, S. Gutwirth, R. Leenes, Y. Poullet (Eds.), European Data Protection: Coming of Age, Springer, 2013.

[52] D. J. Solove, P. Schwartz, Information Privacy Law, 2014.

[53] J. Sedayao, R. Bhardwaj, N. Gorade, Making big data, privacy, and anonymization work together in the enterprise: Experiences and issues, 2014. doi:10.1109/BigData.Congress.2014.92.

[54] A. Cavoukian, Operationalizing privacy by design: A guide to implementing strong privacy practices, 2012.

[55] L. Nurgalieva, A. Frik, G. Doherty, A narrative review of factors affecting the implementation of privacy and security practices in software development, ACM Comput. Surv. 55 (14s) (jul 2023). doi:10.1145/3589951.
URL https://doi.org/10.1145/3589951

[56] S. Spiekermann, J. Korunovska, M. Langheinrich, Inside the organization: Why privacy and security engineering is a challenge for engineers[40pt], Proceedings of the IEEE PP (2018) 1–16. doi:10.1109/JPROC.2018.2866769.

[57] Facebook's zuckerberg admits mistakes in privacy scandal (2018).
URL https://www.columbiavalleypioneer.com/news/facebooks-zuckerberg-admits-mistakes-in-privacy-scandal/

[58] M. Hansen, M. Jensen, M. Rost, Protection goals for privacy engineering, 2015, pp. 159–166. doi:10.1109/SPW.2015.13.

[59] M. Alshammari, A. C. Simpson, Towards a principled approach for engineering privacy by design, in: APF, 2017.

[60] M. Alshammari, A. Simpson, Towards an Effective Privacy Impact and Risk Assessment Methodology: Risk Analysis: ESORICS 2018 International Workshops, DPM 2018 and CBT 2018, Barcelona, Spain, September 6-7, 2018, Proceedings, 2018, pp. 209–224. doi:10.1007/978-3-030-00305-0_16.

[61] R. Gellert, Understanding the notion of risk in the general data protection regulation, Computer Law & Security Review 34 (2) (2018) 279–288. doi:https://doi.org/10.1016/j.clsr.2017.12.003.
URL https://www.sciencedirect.com/science/article/pii/S0267364917302698

[62] S. J. De, D. L. Métayer, Priam: A privacy risk analysis methodology, in: DPM/QASA@ESORICS, 2016.

[63] S. S. Al-Fedaghi, Privacy things: Systematic approach to privacy and personal identifiable information, CoRR abs/1803.10060 (2018). arXiv:1803.10060.
URL http://arxiv.org/abs/1803.10060

[64] K. Wuyts, W. Joosen, Linddun privacy threat modeling: a tutorial, CW Reports (2015).

[65] Y. S. Martín, A. Kung, Methods and tools for gdpr compliance through privacy and data protection engineering, 2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW) (2018) 108–111.

[66] J.-H. Hoepman, Privacy design strategies, in: N. Cuppens-Boulahia, F. Cuppens, S. Jajodia, A. Abou El Kalam, T. Sans (Eds.), ICT Systems Security and Privacy Protection, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014, pp. 446–459.

[67] M. Colesky, J. C. Caiza, J. M. Del Álamo, J.-H. Hoepman, Y.-S. Martín, A system of privacy patterns for user control, in: Proceedings of the 33rd Annual ACM Symposium on Applied Computing, SAC '18, Association for Computing Machinery, New York, NY, USA, 2018, p. 1150–1156. doi:10.1145/3167132.3167257.
URL https://doi.org/10.1145/3167132.3167257

[68] M. Colesky, J. Hoepman, C. Hillen, A critical analysis of privacy design strategies, in: 2016 IEEE Security and Privacy Workshops (SPW), 2016, pp. 33–40. doi:10.1109/SPW.2016.23.

[69] C. Thomborson, Privacy patterns (2016). arXiv:1612.01553.

[70] T. Suphakul, T. Senivongse, Development of privacy design patterns based on privacy principles and uml, in: 2017 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2017, pp. 369–375. doi:10.1109/SNPD.2017.8022748.

[71] A. M. Fernández-Sáez, M. Genero, M. R. V. Chaudron, Does the level of detail of uml models affect the maintainability of source code?, in: EESSMod, 2011.

[72] V. Diamantopoulou, N. Argyropoulos, C. Kalloniatis, S. Gritzalis, Supporting the design of privacy-aware business processes via privacy process patterns, in: 2017 11th International Conference on Research Challenges in Information Science (RCIS), 2017, pp. 187–198. doi:10.1109/RCIS.2017.7956536.

[73] G. Van Blarkom, J. J. Borking, J. E. Olk, Handbook of privacy and privacy-enhancing technologies, Privacy Incorporated Software Agent (PISA) Consortium, The Hague 198 (2003) 14.

[74] M. T. Baldassarre, V. S. Barletta, D. Caivano, M. Scalera, Privacy oriented software development, in: Quality of Information and Communications Technology: 12th International Conference, QUATIC 2019, Ciudad Real, Spain, September 11–13, 2019, Proceedings 12, Springer, 2019, pp. 18–32.

[75] N. Notario, A. Crespo, A. Kung, I. Kroener, D. L. Métayer, C. Troncoso, J. M. del Álamo, Y. S. Martín, Pripare: A new vision on engineering privacy and security by design, in: CSP Forum, 2014.

[76] S. Spiekermann, L. Cranor, Engineering privacy, Software Engineering, IEEE Transactions on 35 (2009) 67 – 82. doi:10.1109/TSE.2008.88.

[77] K. Beckers, S. Faßbender, M. Heisel, R. Meis, A problem-based approach for computer-aided privacy threat identification, in: B. Preneel, D. Ikonomou (Eds.), Privacy Technologies and Policy, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014, pp. 1–16.

[78] R. Meis, Problem-based consideration of privacy-relevant domain knowledge, 2014, pp. 150–164. doi:10.1007/978-3-642-55137-6_12.

[79] Q. He, A. Antón, A framework for modeling privacy requirements in role engineering, Proceedings of the 9th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'03) (06 2003).

[80] D. Ferraris, C. Fernandez-Gago, J. Lopez, A model-driven approach to ensure trust in the iot, Human-centric Computing and Information Sciences 10 (50) (12 2020). doi:10.1186/s13673-020-00257-3.

[81] S. Abu-Nimeh, N. R. Mead, Combining privacy and security risk assessment in security quality requirements engineering, in: AAAI Spring Symposium: Intelligent Information Privacy Management, 2010.

[82] M. N. A. Mohammad, M. Nazir, K. Mustafa, A systematic review and analytical evaluation of security requirements engineering approaches, Arabian Journal for Science and Engineering 44 (11) (2019) 8963–8987.

[83] N. Mead, Adapting the square method for security requirements engineering to acquisition (05 2012).

[84] Cyber Law, Privacy, and Security: Concepts, Methodologies, Tools, and Applications, 2019.

[85] A. Lai, C. Zhang, S. Busovaca, 2-square: A web-based enhancement of square privacy and security requirements engineering, International Journal of Software Innovation 1 (2015) 41–53. doi:10.4018/ijsi.2013010104.

[86] M. T. Ansari, A. Baz, H. Alhakami, W. Alhakami, R. Kumar, P. R. Khan, P-store: Extension of store methodology to elicit privacy requirements, Arabian Journal for Science and Engineering 46 (03 2021). doi:10.1007/s13369-021-05476-z.

[87] C. Kalloniatis, E. Kavakli, S. Gritzalis, Addressing privacy requirements in system design: The pris method, Requir. Eng. 13 (2008) 241–255. doi:10.1007/s00766-008-0067-3.

[88] C. Jensen, J. Tullio, C. Potts, E. D. Mynatt, Strap: a structured analysis framework for privacy, Tech. rep., Georgia Institute of Technology (2005).

[89] R. Meis, Problem-based privacy analysis (propan) – a computer-aided privacy requirements engineering method, Ph.D. thesis (12 2018).

[90] Microsoft threat modeling tool threats.
URL `https://docs.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-threats`

[91] E. Yu, L. Liu, Modelling trust in the i* strategic actors framework (05 2000).

[92] N. Zannone, The si* modeling framework: metamodel and applications, International Journal of Software Engineering and Knowledge Engineering 19 (05) (2009) 727–746.

[93] P. Giorgini, M. Kolp, J. Mylopoulos, M. Pistore, The tropos methodology: An overview, Methodologies and software engineering for agent systems: the agent-oriented software engineering handbook (2004) 89–106.

[94] D. Mellado, E. Fernández-Medina, M. Piattini, Secure tropos framework for software product lines requirements engineering, Information and Software Technology 52 (2010) 1094–1117. `doi:10.1016/j.infsof.2010.05.007`.

[95] M. Pavlidis, H. Mouratidis, E. Panaousis, N. Argyropoulos, Selecting security mechanisms in secure tropos, 2017, pp. 99–114. `doi:10.1007/978-3-319-64483-7_7`.

[96] R. Matulevicius, Security Risk-Aware Secure Tropos, 2017, pp. 77–91. `doi:10.1007/978-3-319-61717-6_6`.

[97] Y. Asnar, P. Giorgini, F. Massacci, N. Zannone, From trust to dependability through risk analysis, 2007. `doi:10.1109/ARES.2007.93`.

[98] I. Alexander, Misuse cases: use cases with hostile intent, IEEE Software 20 (1) (2003) 58–66. `doi:10.1109/MS.2003.1159030`.

[99] M. Jackson, Problem Frames: Analyzing and structuring software development problems, Addison-Wesley Longman Publishing Co., Inc., 2000.

[100] D. Hatebur, M. Heisel, H. Schmidt, Security engineering using problem frames, in: International Conference on Emerging Trends in Information and Communication Security, Springer, 2006, pp. 238–253.

[101] D. Hatebur, M. Heisel, H. Schmidt, A security engineering process based on patterns, 2007, pp. 734–738. `doi:10.1109/DEXA.2007.36`.

[102] AIHLEG, HIGH-LEVEL EXPERT GROUP ON ARTIFICIAL INTELLIGENCE SET UP BY THE EUROPEAN COMMISSION ETHICS GUIDELINES FOR TRUSTWORTHY AI.
URL https://ec.europa.eu/digital

[103] Ai act.
URL https://www.europarl.europa.eu/news/en/headlines/society/20230601STO93804/eu-ai-act-first-regulation-on-artificial-intelligence

[104] M. Veale, R. Binns, L. Edwards, Algorithms that remember: model inversion attacks and data protection law, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 376 (2133) (2018) 20180083. doi:10.1098/rsta.2018.0083.
URL https://doi.org/10.1098%2Frsta.2018.0083

[105] B. Liu, M. Ding, S. Shaham, W. Rahayu, F. Farokhi, Z. Lin, When machine learning meets privacy: A survey and outlook (2020). doi:10.48550/ARXIV.2011.11819.
URL https://arxiv.org/abs/2011.11819

[106] R. Shokri, M. Stronati, C. Song, V. Shmatikov, Membership inference attacks against machine learning models, in: 2017 IEEE symposium on security and privacy (SP), IEEE, 2017, pp. 3–18.

[107] G. M. Ruiz de Arcaute, J. A. Hernández, P. Reviriego, Assessing the impact of membership inference attacks on classical machine learning algorithms, in: 2022 18th International Conference on the Design of Reliable Communication Networks (DRCN), 2022, pp. 1–4. doi:10.1109/DRCN53993.2022.9758025.

[108] M. Nasr, R. Shokri, A. Houmansadr, Machine learning with membership privacy using adversarial regularization, in: Proceedings of the 2018 ACM SIGSAC conference on computer and communications security, 2018, pp. 634–646.

[109] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, M. Backes, Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models, arXiv preprint arXiv:1806.01246 (2018).

[110] J. Jia, A. Salem, M. Backes, Y. Zhang, N. Z. Gong, Memguard: Defending against black-box membership inference attacks via adversarial examples, in: Proceedings of the 2019 ACM SIGSAC conference on computer and communications security, 2019, pp. 259–274.

[111] Z. Yang, B. Shao, B. Xuan, E.-C. Chang, F. Zhang, Defending model inversion and membership inference attacks via prediction purification, arXiv preprint arXiv:2005.03915 (2020).

[112] B. Ghazi, N. Golowich, R. Kumar, P. Manurangsi, C. Zhang, Deep learning with label differential privacy, in: M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, J. W. Vaughan (Eds.), Advances in Neural Information Processing Systems, Vol. 34, Curran Associates, Inc., 2021, pp. 27131–27145.
URL https://proceedings.neurips.cc/paper/2021/file/e3a54649aeec04cf1c13907bc6c5c8aa-Paper.pdf

[113] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, L. Zhang, Deep learning with differential privacy, CCS '16, Association for Computing Machinery, New York, NY, USA, 2016, p. 308–318. doi:10.1145/2976749.2978318.
URL https://doi.org/10.1145/2976749.2978318

[114] L. T. Phong, T. T. Phuong, Privacy-preserving deep learning via weight transmission, IEEE Transactions on Information Forensics and Security 14 (11) (2019) 3003–3015. doi:10.1109/TIFS.2019.2911169.

[115] R. Shokri, V. Shmatikov, Privacy-preserving deep learning, in: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, CCS '15, Association for Computing Machinery, New York, NY, USA, 2015, p. 1310–1321. doi:10.1145/2810103.2813687.
URL https://doi.org/10.1145/2810103.2813687

[116] M. Heikkilä, E. Lagerspetz, S. Kaski, K. Shimizu, S. Tarkoma, A. Honkela, Differentially private bayesian learning on distributed data, Advances in neural information processing systems 30 (2017).

[117] L. Zhao, Q. Wang, Q. Zou, Y. Zhang, Y. Chen, Privacy-preserving collaborative deep learning with unreliable participants, IEEE Transactions on Information Forensics and Security 15 (2019) 1486–1500.

[118] B. Jayaraman, L. Wang, D. Evans, Q. Gu, Distributed learning without distress: Privacy-preserving empirical risk minimization, Advances in Neural Information Processing Systems 31 (2018).

[119] N. Papernot, M. Abadi, U. Erlingsson, I. Goodfellow, K. Talwar, Semi-supervised knowledge transfer for deep learning from private training data, arXiv preprint arXiv:1610.05755 (2016).

[120] M. Kim, O. Günlü, R. F. Schaefer, Federated learning with local differential privacy: Trade-offs between privacy, utility, and communication, in: ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2021, pp. 2650–2654.

[121] D. Ye, S. Shen, T. Zhu, B. Liu, W. Zhou, One parameter defense—defending against data inference attacks via differential privacy, IEEE Transactions on Information Forensics and Security 17 (2022) 1466–1480. doi:10.1109/TIFS.2022.3163591.

[122] M. Fredrikson, S. Jha, T. Ristenpart, Model inversion attacks that exploit confidence information and basic countermeasures, in: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, CCS '15, Association for Computing Machinery, New York, NY, USA, 2015, p. 1322–1333. doi:10.1145/2810103.2813677.
URL https://doi.org/10.1145/2810103.2813677

[123] G. Ateniese, L. V. Mancini, A. Spognardi, A. Villani, D. Vitali, G. Felici, Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers, International Journal of Security and Networks 10 (3) (2015) 137–150. doi:10.1504/IJSN.2015.071829.

[124] Q. Wang, D. Kurz, Reconstructing training data from diverse ml models by ensemble inversion, in: 2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2022, pp. 3870–3878. doi:10.1109/WACV51458.2022.00392.

[125] M. Khosravy, K. Nakamura, Y. Hirose, N. Nitta, N. Babaguchi, Model inversion attack by integration of deep generative models: Privacy-sensitive face generation from a face recognition system, IEEE Transactions on Information Forensics and Security 17 (2022) 357–372. doi:10.1109/TIFS.2022.3140687.

[126] M. Sato, J. Suzuki, H. Shindo, Y. Matsumoto, Interpretable adversarial perturbation in input embedding space for text, in: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18, International Joint Conferences on Artificial Intelligence Organization, 2018, pp. 4323–4330. doi:10.24963/ijcai.2018/601.
URL https://doi.org/10.24963/ijcai.2018/601

[127] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, A. Swami, The limitations of deep learning in adversarial settings, in: 2016 IEEE European Symposium on Security and Privacy (EuroS&P), 2016, pp. 372–387. doi:10.1109/EuroSP.2016.36.

[128] I. Corona, G. Giacinto, F. Roli, Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues, Information Sciences 239 (2013) 201–225. doi:https://doi.org/10.1016/j.ins.2013.03.022.
URL https://www.sciencedirect.com/science/article/pii/S0020025513002119

[129] K. Ren, T. Zheng, Z. Qin, X. Liu, Adversarial attacks and defenses in deep learning, Engineering 6 (3) (2020) 346–360.

[130] A. Qayyum, J. Qadir, M. Bilal, A. Al-Fuqaha, Secure and robust machine learning for healthcare: A survey, IEEE Reviews in Biomedical Engineering PP (2020) 1–1. doi:10.1109/RBME.2020.3013489.

[131] I. J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples (2014). doi:10.48550/ARXIV.1412.6572.
URL https://arxiv.org/abs/1412.6572

[132] T. Tanay, L. Griffin, A boundary tilting persepective on the phenomenon of adversarial examples (2016). doi:10.48550/ARXIV.1608.07690.
URL https://arxiv.org/abs/1608.07690

[133] L. Schmidt, S. Santurkar, D. Tsipras, K. Talwar, A. Mądry, Adversarially robust generalization requires more data (2018). doi:10.48550/ARXIV.1804.11285.
URL https://arxiv.org/abs/1804.11285

[134] S. Bubeck, E. Price, I. Razenshteyn, Adversarial examples from computational constraints (2018). doi:10.48550/ARXIV.1805.10204.
URL https://arxiv.org/abs/1805.10204

[135] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, A. Madry, Adversarial examples are not bugs, they are features (2019). doi:10.48550/ARXIV.1905.02175.
URL https://arxiv.org/abs/1905.02175

[136] B. Biggio, F. Roli, Wild patterns: Ten years after the rise of adversarial machine learning, Pattern Recognition 84 (2018) 317–331. doi:10.1016/j.patcog.2018.07.023.
URL https://doi.org/10.1016%2Fj.patcog.2018.07.023

[137] P.-Y. Chen, Y. Sharma, H. Zhang, J. Yi, C.-J. Hsieh, Ead: Elastic-net attacks to deep neural networks via adversarial examples (2017). doi:10.48550/ARXIV.1709.04114.
URL https://arxiv.org/abs/1709.04114

[138] N. Carlini, D. Wagner, Adversarial examples are not easily detected: Bypassing ten detection methods (2017). doi:10.48550/ARXIV.1705.07263.
URL https://arxiv.org/abs/1705.07263

[139] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks (2017). doi:10.48550/ARXIV.1706.06083.
URL https://arxiv.org/abs/1706.06083

[140] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, C.-J. Hsieh, ZOO, in: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, ACM, 2017. doi:10.1145/3128572.3140448.
URL https://doi.org/10.1145%2F3128572.3140448

[141] J. Uesato, B. O'Donoghue, A. v. d. Oord, P. Kohli, Adversarial risk and the dangers of evaluating against weak attacks (2018). doi:10.48550/ARXIV.1802.05666.
URL https://arxiv.org/abs/1802.05666

[142] A. Ilyas, L. Engstrom, A. Athalye, J. Lin, Black-box adversarial attacks with limited queries and information (2018). doi:10.48550/ARXIV.1804.08598.
URL https://arxiv.org/abs/1804.08598

[143] W. Brendel, J. Rauber, M. Bethge, Decision-based adversarial attacks: Reliable attacks against black-box machine learning models (2017). doi:10.48550/ARXIV.1712.04248.
URL https://arxiv.org/abs/1712.04248

[144] L. Engstrom, B. Tran, D. Tsipras, L. Schmidt, A. Madry, Exploring the landscape of spatial robustness (2017). doi:10.48550/ARXIV.1712.02779.
URL https://arxiv.org/abs/1712.02779

[145] D. Hendrycks, T. G. Dietterich, Benchmarking neural network robustness to common corruptions and surface variations (2018). doi:10.48550/ARXIV.1807.01697.
URL https://arxiv.org/abs/1807.01697

[146] N. Ford, J. Gilmer, N. Carlini, D. Cubuk, Adversarial examples are a natural consequence of test error in noise (2019). doi:10.48550/ARXIV.1901.10513.
URL https://arxiv.org/abs/1901.10513

[147] G. Katz, C. Barrett, D. Dill, K. Julian, M. Kochenderfer, Reluplex: An efficient smt solver for verifying deep neural networks (2017). doi:10.48550/ARXIV.1702.01135.
URL https://arxiv.org/abs/1702.01135

[148] X. Huang, M. Kwiatkowska, S. Wang, M. Wu, Safety verification of deep neural networks (2016). doi:10.48550/ARXIV.1610.06940.
URL https://arxiv.org/abs/1610.06940

[149] V. Tjeng, K. Xiao, R. Tedrake, Evaluating robustness of neural networks with mixed integer programming (2017). doi:10.48550/ARXIV.1711.07356.
URL https://arxiv.org/abs/1711.07356

[150] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, P. McDaniel, Ensemble adversarial training: Attacks and defenses (2017). doi:10.48550/ARXIV.1705.07204.
URL https://arxiv.org/abs/1705.07204

[151] T. Na, J. H. Ko, S. Mukhopadhyay, Cascade adversarial machine learning regularized with a unified embedding (2017). doi:10.48550/ARXIV.1708.02582.
URL https://arxiv.org/abs/1708.02582

[152] F. Farnia, J. M. Zhang, D. Tse, Generalizable adversarial training via spectral normalization (2018). doi:10.48550/ARXIV.1811.07457.
URL https://arxiv.org/abs/1811.07457

[153] N. Papernot, P. McDaniel, X. Wu, S. Jha, A. Swami, Distillation as a defense to adversarial perturbations against deep neural networks (2015). doi:10.48550/ARXIV.1511.04508.
URL https://arxiv.org/abs/1511.04508

[154] N. Carlini, D. Wagner, Defensive distillation is not robust to adversarial examples (2016). doi:10.48550/ARXIV.1607.04311.
URL https://arxiv.org/abs/1607.04311

[155] S. Gu, L. Rigazio, Towards deep neural network architectures robust to adversarial examples (2014). doi:10.48550/ARXIV.1412.5068.
URL https://arxiv.org/abs/1412.5068

[156] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, J. Zhu, Defense against adversarial attacks using high-level representation guided denoiser (2017). doi:10.48550/ARXIV.1712.02976.
URL https://arxiv.org/abs/1712.02976

[157] W. Xu, D. Evans, Y. Qi, Feature squeezing: Detecting adversarial examples in deep neural networks, in: Proceedings 2018 Network and Distributed System Security Symposium, Internet Society, 2018. doi:10.14722/ndss.2018.23198.
URL https://doi.org/10.14722%2Fndss.2018.23198

[158] P. Samangouei, M. Kabkab, R. Chellappa, Defense-gan: Protecting classifiers against adversarial attacks using generative models (2018). doi:10.48550/ARXIV.1805.06605.
URL https://arxiv.org/abs/1805.06605

[159] G. K. Dziugaite, Z. Ghahramani, D. M. Roy, A study of the effect of jpg compression on adversarial images (2016). doi:10.48550/ARXIV.1608.00853.
URL https://arxiv.org/abs/1608.00853

[160] Y. Luo, X. Boix, G. Roig, T. Poggio, Q. Zhao, Foveation-based mechanisms alleviate adversarial examples (2015). doi:10.48550/ARXIV.1511.06292.
URL https://arxiv.org/abs/1511.06292

[161] A. Prakash, N. Moran, S. Garber, A. DiLillo, J. Storer, Deflecting adversarial attacks with pixel deflection (2018). doi:10.48550/ARXIV.1801.08926.
URL https://arxiv.org/abs/1801.08926

[162] U. Shaham, J. Garritano, Y. Yamada, E. Weinberger, A. Cloninger, X. Cheng, K. Stanton, Y. Kluger, Defending against adversarial images using basis functions transformations

(2018). doi:10.48550/ARXIV.1803.10840.
URL https://arxiv.org/abs/1803.10840

[163] D. Meng, H. Chen, Magnet: a two-pronged defense against adversarial examples (2017). doi:10.48550/ARXIV.1705.09064.
URL https://arxiv.org/abs/1705.09064

[164] J. H. Metzen, T. Genewein, V. Fischer, B. Bischoff, On detecting adversarial perturbations (2017). doi:10.48550/ARXIV.1702.04267.
URL https://arxiv.org/abs/1702.04267

[165] K. Grosse, P. Manoharan, N. Papernot, M. Backes, P. McDaniel, On the (statistical) detection of adversarial examples (2017). doi:10.48550/ARXIV.1702.06280.
URL https://arxiv.org/abs/1702.06280

[166] X. Li, F. Li, Adversarial examples detection in deep networks with convolutional filter statistics (2016). doi:10.48550/ARXIV.1612.07767.
URL https://arxiv.org/abs/1612.07767

[167] J. Lu, T. Issaranon, D. Forsyth, Safetynet: Detecting and rejecting adversarial examples robustly (2017). doi:10.48550/ARXIV.1704.00103.
URL https://arxiv.org/abs/1704.00103

[168] H. Kannan, A. Kurakin, I. Goodfellow, Adversarial logit pairing (2018). doi:10.48550/ARXIV.1803.06373.
URL https://arxiv.org/abs/1803.06373

[169] H. Hosseini, Y. Chen, S. Kannan, B. Zhang, R. Poovendran, Blocking transferability of adversarial examples in black-box learning systems (2017). doi:10.48550/ARXIV.1703.04318.
URL https://arxiv.org/abs/1703.04318

[170] C. Dunn, N. Moustafa, B. Turnbull, Robustness evaluations of sustainable machine learning models against data poisoning attacks in the internet of things, Sustainability 12 (16) (2020). doi:10.3390/su12166434.
URL https://www.mdpi.com/2071-1050/12/16/6434

[171] O. Alvear, C. T. Calafate, J.-C. Cano, P. Manzoni, Crowdsensing in smart cities: Overview, platforms, and environment sensing issues, Sensors 18 (2) (2018).
URL https://www.mdpi.com/1424-8220/18/2/460

[172] M. Li, Y. Sun, H. Lu, S. Maharjan, Z. Tian, Deep reinforcement learning for partially observable data poisoning attack in crowdsensing systems, IEEE Internet of Things Journal 7 (7) (2019) 6266–6278.

[173] Z. Huang, M. Pan, Y. Gong, Robust truth discovery against data poisoning in mobile crowdsensing, in: 2019 IEEE Global Communications Conference (GLOBECOM), IEEE, 2019, pp. 1–6.

[174] C. Miao, Q. Li, H. Xiao, W. Jiang, M. Huai, L. Su, Towards data poisoning attacks in crowd sensing systems, in: Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing, Mobihoc '18, Association for Computing Machinery, New York, NY, USA, 2018, p. 111–120. doi:10.1145/3209582.3209594.
URL https://doi.org/10.1145/3209582.3209594

[175] C. Miao, Q. Li, L. Su, M. Huai, W. Jiang, J. Gao, Attack under disguise: An intelligent data poisoning attack mechanism in crowdsourcing, 2018, pp. 13–22. doi:10.1145/3178876.3186032.

[176] A. I. Newaz, N. I. Haque, A. K. Sikder, M. A. Rahman, A. S. Uluagac, Adversarial attacks to machine learning-based smart healthcare systems (2020). doi:10.48550/ARXIV.2010.03671.
URL https://arxiv.org/abs/2010.03671

[177] N. N. Dalvi, P. M. Domingos, Mausam, S. K. Sanghai, D. Verma, Adversarial classification, Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining (2004).

[178] D. Lowd, C. Meek, Good word attacks on statistical spam filters, in: CEAS, 2005.

[179] B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. P. Rubinstein, U. Saini, C. Sutton, J. D. Tygar, K. Xia, Exploiting machine learning to subvert your spam filter, in: Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats, LEET'08, USENIX Association, USA, 2008.

[180] B. Biggio, B. Nelson, P. Laskov, Poisoning attacks against support vector machines (2012). doi:10.48550/ARXIV.1206.6389.
URL https://arxiv.org/abs/1206.6389

[181] J. Steinhardt, P. W. Koh, P. Liang, Certified defenses for data poisoning attacks (2017). doi:10.48550/ARXIV.1706.03691.
URL https://arxiv.org/abs/1706.03691

[182] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, F. Roli, Towards poisoning of deep learning algorithms with back-gradient optimization (2017). doi:10.48550/ARXIV.1708.08689.
URL https://arxiv.org/abs/1708.08689

[183] C. Yang, Q. Wu, H. Li, Y. Chen, Generative poisoning attack method against neural networks (2017). doi:10.48550/ARXIV.1703.01340.
URL https://arxiv.org/abs/1703.01340

[184] T. Gu, K. Liu, B. Dolan-Gavitt, S. Garg, Badnets: Evaluating backdooring attacks on deep neural networks, IEEE Access 7 (2019) 47230–47244. doi:10.1109/ACCESS.2019.2909068.

[185] B. Biggio, K. Rieck, D. Ariu, C. Wressnegger, I. Corona, G. Giacinto, F. Roli, Poisoning behavioral malware clustering (2018). doi:10.48550/ARXIV.1811.09985.
URL https://arxiv.org/abs/1811.09985

[186] R. Perdisci, D. Dagon, W. Lee, P. Fogla, M. Sharif, Misleading worm signature generators using deliberate noise injection, in: 2006 IEEE Symposium on Security and Privacy (S&P'06), 2006, pp. 15 pp.–31. doi:10.1109/SP.2006.26.

[187] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, B. Li, Manipulating machine learning: Poisoning attacks and countermeasures for regression learning (2018). doi:10.48550/ARXIV.1804.00308.
URL https://arxiv.org/abs/1804.00308

[188] A. Paudice, L. Muñoz-González, A. Gyorgy, E. C. Lupu, Detection of adversarial training examples in poisoning attacks through anomaly detection (2018). doi:10.48550/ARXIV.1802.03041.
URL https://arxiv.org/abs/1802.03041

[189] B. Nelson, M. Barreno, F. Jack Chi, A. D. Joseph, B. I. P. Rubinstein, U. Saini, C. Sutton, J. D. Tygar, K. Xia, Misleading Learners: Co-opting Your Spam Filter, Springer US, Boston, MA, 2009, pp. 17–51. doi:10.1007/978-0-387-88735-7_2.
URL https://doi.org/10.1007/978-0-387-88735-7_2

[190] O. Suciu, R. Mărginean, Y. Kaya, H. Daumé, T. Dumitraş, When does machine learning fail? generalized transferability for evasion and poisoning attacks, SEC'18, USENIX Association, USA, 2018, p. 1299–1316.

[191] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, S. Nepal, Strip: A defence against trojan attacks on deep neural networks (2019). doi:10.48550/ARXIV.1902.06531.
URL https://arxiv.org/abs/1902.06531

[192] S. Mei, X. Zhu, Using machine teaching to identify optimal training-set attacks on machine learners, in: AAAI, 2015.

[193] Y. Liu, A. Mondal, A. Chakraborty, M. Zuzak, N. Jacobsen, D. Xing, A. Srivastava, A survey on neural trojans, in: 2020 21st International Symposium on Quality Electronic Design (ISQED), IEEE, 2020, pp. 33–39.

[194] T. Gu, B. Dolan-Gavitt, S. Garg, Badnets: Identifying vulnerabilities in the machine learning model supply chain (2017). doi:10.48550/ARXIV.1708.06733.
URL https://arxiv.org/abs/1708.06733

[195] A. Salem, R. Wen, M. Backes, S. Ma, Y. Zhang, Dynamic backdoor attacks against machine learning models (2020). doi:10.48550/ARXIV.2003.03675.
URL https://arxiv.org/abs/2003.03675

[196] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, X. Zhang, Trojaning attack on neural networks, in: NDSS, 2018.

[197] Y. Yao, H. Li, H. Zheng, B. Y. Zhao, Latent backdoor attacks on deep neural networks, in: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 2041–2055. doi:10.1145/3319535.3354209.
URL https://doi.org/10.1145/3319535.3354209

[198] S. Kaviani, I. Sohn, Defense against neural trojan attacks: A survey, Neurocomputing 423 (2021) 651–667. doi:https://doi.org/10.1016/j.neucom.2020.07.133.
URL https://www.sciencedirect.com/science/article/pii/S0925231220316350

[199] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, F. Roli, Evasion attacks against machine learning at test time, in: H. Blockeel, K. Kersting, S. Nijssen, F. Železný (Eds.), Machine Learning and Knowledge Discovery in Databases, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 387–402.

[200] L. Demetrio, B. Biggio, G. Lagorio, F. Roli, A. Armando, Explaining vulnerabilities of deep learning to adversarial malware binaries (2019). doi:10.48550/ARXIV.1901.03583.
URL https://arxiv.org/abs/1901.03583

[201] M. Juuti, S. Szyller, S. Marchal, N. Asokan, Prada: Protecting against dnn model stealing attacks, in: 2019 IEEE European Symposium on Security and Privacy (EuroS&P), 2019, pp. 512–527. doi:10.1109/EuroSP.2019.00044.

[202] K. Krishna, G. S. Tomar, A. P. Parikh, N. Papernot, M. Iyyer, Thieves on sesame street! model extraction of bert-based apis (2019). doi:10.48550/ARXIV.1910.12366.
URL https://arxiv.org/abs/1910.12366

[203] T. Orekondy, B. Schiele, M. Fritz, Prediction poisoning: Towards defenses against dnn model stealing attacks, arXiv preprint arXiv:1906.10908 (2019).

[204] D. Lowd, C. Meek, Adversarial learning, KDD '05, Association for Computing Machinery, New York, NY, USA, 2005, p. 641–647. doi:10.1145/1081870.1081950.
URL https://doi.org/10.1145/1081870.1081950

[205] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, T. Ristenpart, Stealing machine learning models via prediction apis (2016). doi:10.48550/ARXIV.1609.02943.
URL https://arxiv.org/abs/1609.02943

[206] S. J. Oh, M. Augustin, B. Schiele, M. Fritz, Towards reverse-engineering black-box neural networks (2017). doi:10.48550/ARXIV.1711.01768.
URL https://arxiv.org/abs/1711.01768

[207] B. Wang, N. Z. Gong, Stealing hyperparameters in machine learning, in: 2018 IEEE Symposium on Security and Privacy (SP), 2018, pp. 36–52. doi:10.1109/SP.2018.00038.

[208] W. Hua, Z. Zhang, G. E. Suh, Reverse engineering convolutional neural networks through side-channel information leaks, DAC '18, Association for Computing Machinery, New York, NY, USA, 2018. doi:10.1145/3195970.3196105.
URL https://doi.org/10.1145/3195970.3196105

[209] Y. Wang, D. J. Miller, G. Kesidis, When not to classify: Detection of reverse engineering attacks on dnn image classifiers, in: ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019, pp. 8063–8066. doi:10.1109/ICASSP.2019.8682578.

[210] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, et al., Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai, Information fusion 58 (2020) 82–115.

[211] C. O'neil, Weapons of math destruction: How big data increases inequality and threatens democracy, Broadway books, 2016.

[212] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, D. Pedreschi, A survey of methods for explaining black box models, ACM Comput. Surv. 51 (5) (08 2018). doi:10.1145/3236009.
URL https://doi.org/10.1145/3236009

[213] I. Kamwa, S. Samantaray, G. Joós, On the accuracy versus transparency trade-off of data-mining models for fast-response pmu-based catastrophe predictors, IEEE Transactions on Smart Grid 3 (1) (2011) 152–161.

[214] A. Ignatiev, N. Narodytska, J. Marques-Silva, On relating explanations and adversarial examples, Advances in neural information processing systems 32 (2019).

[215] G. Fidel, R. Bitton, A. Shabtai, When explainability meets adversarial learning: Detecting adversarial examples using shap signatures (2019). doi:10.48550/ARXIV.1909.03418.
URL https://arxiv.org/abs/1909.03418

[216] P. Panda, K. Roy, Explainable adversarial learning: Implicit generative modeling of random noise during training for adversarial robustness (2018).

[217] S. M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, Advances in neural information processing systems 30 (2017).

[218] J. Dhaliwal, S. Shintre, Gradient similarity: An explainable approach to detect adversarial attacks against deep learning (2018). doi:10.48550/ARXIV.1806.10707.
URL https://arxiv.org/abs/1806.10707

[219] D. L. Marino, C. S. Wickramasinghe, M. Manic, An adversarial approach for explainable ai in intrusion detection systems, in: IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society, 2018, pp. 3237–3243. doi:10.1109/IECON.2018.8591457.

[220] A. Rahnama, A. Tseng, An adversarial approach for explaining the predictions of deep neural networks (2020). doi:10.48550/ARXIV.2005.10284.
URL https://arxiv.org/abs/2005.10284

[221] A. Michel, S. K. Jha, R. Ewetz, A survey on the vulnerability of deep neural networks against adversarial attacks, Progress in Artificial Intelligence 11 (2) (2022) 131 – 141, cited by: 1. doi:10.1007/s13748-021-00269-9.
URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85123097338&doi=10.1007%2fs13748-021-00269-9&partnerID=40&md5=b9a01a1f7c0bd3abef886f603e46bc89

[222] A. Agarwal, A. Beygelzimer, M. Dudik, J. Langford, H. Wallach, A reductions approach to fair classification, in: J. Dy, A. Krause (Eds.), Proceedings of the 35th International Conference on Machine Learning, Vol. 80 of Proceedings of Machine Learning Research, PMLR, 2018, pp. 60–69.

[223] T. Calders, F. Kamiran, M. Pechenizkiy, Building classifiers with independency constraints, in: 2009 IEEE International Conference on Data Mining Workshops, 2009, pp. 13–18.

[224] M. Hardt, E. Price, E. Price, N. Srebro, Equality of opportunity in supervised learning, in: D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 29, Curran Associates, Inc., 2016.
URL https://proceedings.neurips.cc/paper_files/paper/2016/file/9d2682367c3935defcb1f9e247a97c0d-Paper.pdf

[225] S. Corbett-Davies, E. Pierson, A. Feller, S. Goel, A. Huq, Algorithmic decision making and the cost of fairness, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 797–806. doi:10.1145/3097983.3098095.
URL https://doi.org/10.1145/3097983.3098095

[226] S. o. I. UC Berkeley, Privacy patterns.
URL https://privacypatterns.org

[227] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann, P. Sommerlad, Security Patterns: Integrating security and systems engineering, John Wiley & Sons, 2013.

[228] O. Foundation, Top 10 web application security risk, https://owasp.org/www-project-top-ten/ (2021).

[229] M. T. Baldassarre, V. S. Barletta, G. Dimauro, D. Gigante, A. Pagano, A. Piccinno, Supporting secure agile development: the vis-prise tool, in: Proceedings of the 2022 International Conference on Advanced Visual Interfaces, AVI 2022, Association for Computing Machinery, New York, NY, USA, 2022. doi:10.1145/3531073.3534494.
URL https://doi.org/10.1145/3531073.3534494

[230] V. R. Basili, G. Caldiera, H. D. Rombach, The goal question metric approach, 1994.
URL https://api.semanticscholar.org/CorpusID:13884048

[231] M. Sobolewski, J. Mazur, M. Paliński, Gdpr: A step towards a user-centric internet?, Intereconomics 52 (2017) 207–213. doi:10.1007/s10272-017-0676-5.

[232] K. Hjerppe, J. Ruohonen, V. Leppänen, The general data protection regulation: Requirements, architectures, and constraints (07 2019).

[233] T. Jakobi, S. Patil, D. Randall, G. Stevens, V. Wulf, It is about what they could do with the data: A user perspective on privacy in smart metering, ACM Trans. Comput.-Hum. Interact. 26 (1) (jan 2019). doi:10.1145/3281444.
URL https://doi.org/10.1145/3281444

[234] S. Alpers, A. Oberweis, M. Pieper, S. Betz, A. Fritsch, G. Schiefer, M. Wagner, Privacy-avare: An approach to manage and distribute privacy settings, 2017. doi:10.1109/CompComm.2017.8322784.

[235] H. M. Sneed, Softwaremanagement, Verlagsgesellschaft Rudolf Müller, 1987.

[236] T. Ōno, Toyota Production System: Beyond Large-Scale Production, Productivity Press, 1988.

[237] J. P. Womack, D. T. Jones, Lean Thinking: Banish Waste and Create Wealth in Your Corporation, 2nd Edition, Free Press, 2003.

[238] A. Janes, B. Russo, Automatic Performance Monitoring and Regression Testing During the Transition from Monolith to Microservices, in: International Symposium on Software Reliability Engineering Workshops (ISSREW), 2019, pp. 163–168. doi:10.1109/ISSREW.2019.00067.

[239] D. Gadler, M. Mairegger, A. Janes, B. Russo, Mining Logs to Model the Use of a System, in: International Symposium on Empirical Software Engineering and Measurement (ESEM), 2017, pp. 334–343. doi:10.1109/ESEM.2017.47.

[240] L. Corral, A. B. Georgiev, A. Janes, S. Kofler, Energy-Aware Performance Evaluation of Android Custom Kernels, in: International Workshop on Green and Sustainable Software, 2015, p. 1–7.

[241] K. Beck, et al., Manifesto for Agile Software Development, http://www.agilemanifesto.org/ (2001).

[242] K. Beck, C. Andres, Extreme Programming Explained: Embrace Change, 2nd Edition, Addison-Wesley, 2004.

[243] N. A. Ernst, S. Bellomo, I. Ozkaya, R. L. Nord, I. Gorton, Measure it? Manage it? Ignore it? Software practitioners and technical debt, Symposium on the Foundations of Software Engineering (2015) 50–60.

[244] C. Vassallo, S. Panichella, F. Palomba, S. Proksch, H. C. Gall, A. Zaidman, How developers engage with static analysis tools in different contexts, Empirical Software Engineering (2019) 1–39.

[245] P. Avgeriou, et al., An Overview and Comparison of Technical Debt Measurement Tools, IEEE Software (2021).

[246] V. Lenarduzzi, N. Saarimäki, D. Taibi, Some SonarQube Issues have a Significant but SmallEffect on Faults and Changes. A large-scale empirical study, Journal of Systems and Software 170 (2020).

[247] F. Zampetti, S. Scalabrino, R. Oliveto, G. Canfora, M. Di Penta, How open source projects use static code analysis tools in continuous integration pipelines, in: Int. Conf. on Mining Software Repositories, 2017, pp. 334–344.

[248] M. Mantere, I. Uusitalo, J. Roning, Comparison of static code analysis tools, in: Int. Conf. on Emerging Security Information, Systems and Technologies, 2009, pp. 15–22.

[249] V. Lenarduzzi, F. Pecorelli, N. Saarimaki, S. Lujan, F. Palomba, A critical comparison on six static analysis tools: Detection, agreement, and precision, Journal of Systems and Software 198 (2023).

[250] R. K. McLean, Comparing static security analysis tools using open source software, in: Int. Conf. on Software Security and Reliability, 2012, pp. 68–74.

[251] M. A. Al Mamun, A. Khanam, H. Grahn, R. Feldt, Comparing four static analysis tools for java concurrency bugs, in: Third Swedish Workshop on Multi-Core Computing (MCC-10), 2010.

[252] F. Pecorelli, S. Lujan, V. Lenarduzzi, F. Palomba, A. De Lucia, On the adequacy of static analysis warnings with respect to code smell prediction, Empirical Software Engineering 27 (3) (2022).

[253] D. Gardner, M. Horvath, D. Zumerle, Magic Quadrant for Application Security Testing (April 2022).

[254] N. Imtiaz, A. Rahman, E. Farhana, L. Williams, Challenges with Responding to Static Analysis Tool Alerts, in: International Conference on Mining Software Repositories (MSR), 2019, pp. 245–249. doi:10.1109/MSR.2019.00049.

[255] M. Christakis, C. Bird, What developers want and need from program analysis: An empirical study, in: International Conference on Automated Software Engineering (ASE), 2016, pp. 332–343.

[256] I. Pashchenko, S. Dashevskyi, F. Massacci, Delta-Bench: Differential Benchmark for Static Analysis Security Testing Tools, in: International Symposium on Empirical Software Engineering and Measurement (ESEM), 2017, pp. 163–168. doi:10.1109/ESEM.2017.24.

[257] M. Dowd, J. McDonald, J. Schuh, The art of software security assessment: Identifying and preventing software vulnerabilities, Pearson Education, 2006.

[258] G. McGraw, Software security, IEEE Security & Privacy 2 (2) (2004) 80–83.

[259] A. Decan, T. Mens, E. Constantinou, On the impact of security vulnerabilities in the npm package dependency network, in: International Conference on Mining Software Repositories, 2018, pp. 181–191.

[260] H. Plate, S. E. Ponta, A. Sabetta, Impact assessment for vulnerabilities in open-source software libraries, in: International Conference on Software Maintenance and Evolution (ICSME), 2015, pp. 411–420.

[261] M. Finifter, D. Akhawe, D. Wagner, An empirical study of vulnerability rewards programs, in: Security Symposium, 2013, pp. 273–288.

[262] S. Kim, H. Lee, Software systems at risk: An empirical study of cloned vulnerabilities in practice, Computers & Security 77 (2018) 720–736.

[263] D. Gonzalez, F. Alhenaki, M. Mirakhorli, Architectural security weaknesses in industrial control systems (ICS) an empirical study based on disclosed software vulnerabilities, in: International Conference on Software Architecture (ICSA), 2019, pp. 31–40.

[264] J. Svacina, et al., On vulnerability and security log analysis: A systematic literature review on recent trends, in: International Conference on Research in Adaptive and Convergent Systems, 2020, pp. 175–180.

[265] M.-T. Trinh, D.-H. Chu, J. Jaffar, S3: A symbolic string solver for vulnerability detection in web applications, in: Conference on Computer and Communications Security, 2014, pp. 1232–1243.

[266] H. Li, T. Kim, M. Bat-Erdene, H. Lee, Software vulnerability detection using backward trace analysis and symbolic execution, in: International Conference on Availability, Reliability and Security, 2013, pp. 446–454.

[267] B. Jiang, Y. Liu, W. Chan, Contractfuzzer: Fuzzing smart contracts for vulnerability detection, in: International Conference on Automated Software Engineering (ASE), IEEE, 2018, pp. 259–269.

[268] T. Wang, T. Wei, G. Gu, W. Zou, TaintScope: A checksum-aware directed fuzzing tool for automatic software vulnerability detection, in: Symposium on Security and Privacy, IEEE, 2010, pp. 497–512.

[269] R. Scandariato, J. Walden, A. Hovsepyan, W. Joosen, Predicting Vulnerable Software Components via Text Mining, Trans. Software Eng. 40 (10) (2014) 993–1006.

[270] V. Lenarduzzi, N. Saarimäki, D. Taibi, Some sonarqube issues have a significant but small effect on faults and changes. a large-scale empirical study, Journal of Systems and Software (2020) 110750.

[271] D. Stefanović, D. Nikolić, D. Dakić, I. Spasojević, S. Ristić, Static code analysis tools: A systematic literature review, International DAAAM Symposium 31 (1) (2020) 565–573.

[272] A. Nguyen-Duc, et al., On the adoption of static analysis for software security assessment–A case study of an open-source e-government project, Computers & Security 111 (2021) 102470.

[273] M. T. Baldassarre, M. Piattini, F. J. Pino, G. Visaggio, Comparing ISO/IEC 12207 and CMMI-DEV: Towards a mapping of ISO/IEC 15504-7, International Conference on Software Engineering (2009).

[274] M. T. Baldassarre, D. Caivano, G. Visaggio, Software Renewal Projects Estimation Using Dynamic Calibration, International Conference on Software Maintenance, ICSM (2003) 105–115.

[275] D. Gigante, F. Pecorelli, V. S. Barletta, A. Janes, V. Lenarduzzi, D. Taibi, M. T. Baldassarre, Sonarqube vs fortify appendix, https://figshare.com/s/f229f6590c38032ebea3 (2023).

[276] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics bulletin 1 (6) (1945) 80–83.

[277] N. Cliff, Dominance statistics: Ordinal analyses to answer ordinal questions., Psychological bulletin 114 (3) (1993) 494.

[278] D. Greene, A. L. Hoffmann, L. Stark, Better, nicer, clearer, fairer: A critical assessment of the movement for ethical artificial intelligence and machine learning, in: Hawaii International Conference on System Sciences, 2019.

[279] Wagner, B. in *Being Profiled: Cogitas Ergo Sum. 10 Years of 'Profiling the European Citizen'* (eds Bayamlioglu, E., Baraliuc, I., Janssens, L. A. W. & Hildebrandt, M.), Amsterdam University Press, 2018. doi:10.1515/9789048550180.
URL https://mediarep.org/handle/doc/14277

[280] L. Floridi, J. Cowls, A unified framework of five principles for ai in society, Issue 1 (2019).

[281] High-Level Expert Group on AI (AIHLEG), Ethics guidelines for trustworthy AI | Shaping Europe's digital future (2018).
URL https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai

[282] R. E. Johnson, Frameworks = (components + patterns), Communications of the ACM 40 (10) (1997) 39 – 42. doi:10.1145/262793.262799.

[283] K. J. Konnyu, E. Kwok, B. Skidmore, D. Moher, The effectiveness and safety of emergency department short stay units: a rapid review 6 (1) (2012) 10–16.
URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3329070/

[284] A. Watt, A. Cameron, L. Sturm, T. Lathlean, W. Babidge, S. Blamey, K. Facey, D. Hailey, I. Norderhaug, G. Maddern, et al., Rapid reviews versus full systematic reviews: An inventory of current methods and practice in health technology assessment, International Journal of Technology Assessment in Health Care 24 (2) (2008) 133–139. doi:10.1017/S0266462308080185.

[285] B. Cartaxo, G. Pinto, S. Soares, The role of rapid reviews in supporting decision-making in software engineering practice, in: Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018, EASE'18, Association for Computing Machinery, New York, NY, USA, 2018, p. 24–34. doi:10.1145/3210459.3210462.
URL https://doi.org/10.1145/3210459.3210462

[286] B. Kitchenham, S. Charters, Guidelines for performing systematic literature reviews in software engineering 2 (01 2007).

[287] J. Piasecki, M. Waligora, V. Dranseika, Google search as an additional source in systematic reviews, Science and Engineering Ethics 24 (12 2017). doi:10.1007/s11948-017-0010-4.

[288] V. S. Barletta, D. Caivano, D. Gigante, A. Ragone, Rapid review online appendix (2023). URL https://figshare.com/s/ed4bc1b40d7ad5d7da31

[289] B. Mittelstadt, Principles alone cannot guarantee ethical ai, Nature Machine Intelligence 1 (11) (2019) 501–507. doi:10.1038/s42256-019-0114-4.
URL https://doi.org/10.1038/s42256-019-0114-4

[290] L. Floridi, Translating principles into practices of digital ethics: Five risks of being unethical, Philosophy & Technology 32 (2) (2019) 185–193. doi:10.1007/s13347-019-00354-x.

[291] M. T. Baldassarre, V. S. Barletta, D. Caivano, M. Scalera, Privacy oriented software development, in: Quality of Information and Communications Technology, 2019.

[292] F. Zhengxin, Y. Yi, Z. Jingyu, L. Yue, M. Yuechen, L. Qinghua, X. Xiwei, W. Jeff, W. Chen, Z. Shuai, C. Shiping, Mlops spanning whole machine learning life cycle: A survey, ArXiv abs/2304.07296 (2023).

[293] D. D. Heckathorn, Comment: Snowball versus respondent-driven sampling, Sociological Methodology 41 (2011) 355 – 366.
URL https://api.semanticscholar.org/CorpusID:117640237

[294] M. T. Baldassarre, D. Gigante, M. Kalinowski, A. Ragone, S. Tibidò, Survey link, https://forms.office.com/e/GVeeWf1Pqz (2024).

[295] M. T. Baldassarre, D. Gigante, M. Kalinowski, A. Ragone, S. Tibidò, Online appendix, https://figshare.com/s/099d7f7a0b2da08653e6 (2024).

[296] J. Kupis, S. Johnson, G. Hallihan, D. L. Olstad, Assessing the usability of the automated self-administered dietary assessment tool (asa24) among low-income adults, Nutrients 11 (1) (2019).

[297] M. Someren, Y. Barnard, J. Sandberg, The Think Aloud Method - A Practical Guide to Modelling CognitiveProcesses, 1994.

[298] D. George, P. Mallery, IBM SPSS Statistics 25 Step by Step: A Simple Guide and Reference, 2018.

[299] D. Cruzes, T. Dybå, Recommended steps for thematic synthesis in software engineering, 2011, pp. 275 – 284. doi:10.1109/ESEM.2011.36.

[300] V. Braun, V. Clarke, Using thematic analysis in psychology, Qualitative Research in Psychology 3 (2006) 77–101.

[301] V. Braun, V. Clarke, Thematic analysis., 2012, pp. 57–71.

[302] R. Bakeman, V. Quera, Observer Agreement and Cohen's Kappa, Cambridge University Press, 2011, p. 57–71.

[303] Y. Benjamini, Y. Hochberg, Controlling the false discovery rate: a practical and powerful approach to multiple testing, Journal of the Royal statistical society: series B (Methodological) 57 (1) (1995) 289–300.

[304] Gemini image generation got it wrong. we'll do better. (2024).
URL {https://blog.google/products/gemini/gemini-image-generation-issue}

[305] European Parliament, Council of the European Union, Regulation (EU) 2016/679 of the European Parliament and of the Council (2016-05-04).
URL https://data.europa.eu/eli/reg/2016/679/oj

[306] S. M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems 30, Curran Associates, Inc., 2017, pp. 4765–4774.

[307] M. T. Ribeiro, S. Singh, C. Guestrin, "why should I trust you?": Explaining the predictions of any classifier, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016, 2016, pp. 1135–1144.

[308] E. M. Kenny, C. Ford, M. Quinn, M. T. Keane, Explaining black-box classifiers using post-hoc explanations-by-example: The effect of explanations and error-rates in xai user studies, Artificial Intelligence 294 (2021) 103459.

[309] European Union, AI Act (2023).
URL https://www.europarl.europa.eu/news/en/headlines/society/20230601STO93804/eu-ai-act-first-regulation-on-artificial-intelligence

[310] W. Jin, J. Fan, D. Gromala, P. Pasquier, G. Hamarneh, Euca: the end-user-centered explainable ai framework (2021). arXiv:2102.02437.

[311] S. Muhammad, The fairness handbook (2022).
URL https://amsterdamintelligence.com/resources/the-fairness-handbook

[312] ENISA, Securing machine learning algorithms (2021).
URL https://www.enisa.europa.eu/publications/securing-machine-learning-algorithms

[313] Tensorflow, Responsible ai in your ml workflow (2021).
URL https://www.tensorflow.org/responsible_ai?hl=en

[314] D. CSIRO, Responsible ai pattern catalogue (2022).
URL https://research.csiro.au/ss/science/projects/responsible-ai-pattern-catalogue/

[315] ICO, Guidance on AI and data protection (2021).
URL https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/artificial-intelligence/guidance-on-ai-and-data-protection/

[316] Microsoft, Threat modeling AI/ML systems and dependencies (2021).
URL https://learn.microsoft.com/en-us/security/engineering/threat-modeling-aiml

[317] A. Wildberger, Alleviating the opacity of neural networks, in: Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94), Vol. 4, 1994, pp. 2373–2376 vol.4. doi:10.1109/ICNN.1994.374590.

[318] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, S.-I. Lee, From local explanations to global understanding with explainable ai for trees, Nature Machine Intelligence 2 (1) (2020) 56–67. doi:10.1038/s42256-019-0138-9.
URL https://doi.org/10.1038/s42256-019-0138-9

[319] P. Runeson, M. Höst, Guidelines for conducting and reporting case study research in software engineering, Empirical Software Engineering 14 (2009) 131–164.
URL https://api.semanticscholar.org/CorpusID:207144526

[320] B. Jahne, Computer vision and applications: a guide for students and practitioners, Elsevier, 2000.

[321] C. Lewis, Using the" thinking-aloud" method in cognitive interface design, IBM TJ Watson Research Center Yorktown Heights, NY, 1982.

[322] H. Kuusela, P. Paul, A comparison of concurrent and retrospective verbal protocol analysis, The American Journal of Psychology 113 (3) (2000) 387–404.
URL http://www.jstor.org/stable/1423365

[323] J. W. Creswell, Research design: Qualitative, quantitative, and mixed methods approaches, Sage, 2013.

[324] M. L. Mohus, J. Li, Adversarial robustness in unsupervised machine learning: A systematic review (2023). arXiv:2306.00687.

[325] M. T. Baldassarre, D. Gigante, M. Kalinowski, A. Ragone, Polaris github link, https://github.com/dom976/POLARIS_framework (2023).

[326] D. Zowghi, C. Coulin, Requirements elicitation: A survey of techniques, approaches, and tools, Engineering and managing software requirements (2005) 19–46.

[327] M. T. Baldassarre, M. Gigante, A. Ragone, Polaris elicitation study interview link, https://forms.office.com/e/KVLXP5iYVv (2024).

[328] M. T. Baldassarre, D. Gigante, M. Kalinowski, A. Ragone, Polaris elicitation study online appendix, https://figshare.com/s/e961d4a41c0887c2e17c (2023).

[329] M. T. Baldassarre, D. Gigante, M. Kalinowski, A. Ragone, Polaris user study survey link, https://forms.office.com/e/2k5D1gGWeE (2024).

[330] M. T. Baldassarre, D. Gigante, M. Kalinowski, A. Ragone, Polaris final user study, https://figshare.com/s/05f0316ceaf6d32ed7bf (2024).

[331] A. M. Lund, Measuring usability with the use questionnaire12, Usability interface 8 (2) (2001) 3–6.

[332] F. D. Davis, et al., Technology acceptance model: Tam, Al-Suqri, MN, Al-Aufi, AS: Information Seeking Behavior and Technology Adoption 205 (1989) 219.

[333] Proposal for a REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL LAYING DOWN HARMONISED RULES ON ARTIFICIAL INTELLIGENCE (ARTIFICIAL INTELLIGENCE ACT) AND AMENDING CERTAIN UNION LEGISLATIVE ACTS (2021).
URL https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex%3A52021PC0206

[334] International Journal of Human Rights Law Review, ChatGPT and human rights: Navigating the technological frontier.
URL https://humanrightlawreview.in/chatgpt-and-human-rights-navigating-the-technological-frontier/

[335] P. P. Ray, ChatGPT: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope 3 121–154. doi:10.1016/j.iotcps.2023.04.003.
URL https://www.sciencedirect.com/science/article/pii/S266734522300024X

[336] J. Paul, A. Ueno, C. Dennis, ChatGPT and consumers: Benefits, pitfalls and future research agenda ijcs.12928doi:10.1111/ijcs.12928.
URL https://onlinelibrary.wiley.com/doi/10.1111/ijcs.12928

[337] D. Hillemann, S. Zimprich, ChatGPT - legal challenges, legal opportunities.
URL https://www.fieldfisher.com/en/insights/chatgpt-legal-challenges-legal-opportunities

[338] J. Davis, ChatGPT: Enterprises eye use cases, ethicists remain concerned.
URL https://www.informationweek.com/big-data/chat-gpt-enterprises-eye-use-cases-ethicists-remain-concerned

[339] S. Lock, What is AI chatbot phenomenon ChatGPT and could it replace humans?
URL https://www.theguardian.com/technology/2022/dec/05/what-is-ai-chatbot-phenomenon-chatgpt-and-could-it-replace-humans

[340] P. Deo, Is ChatGPT dangerous for humanity?
URL https://timesofindia.indiatimes.com/business/india-business/is-chatgpt-dangerous-for-humanity/articleshow/98471105.cms

[341] P. Rivas, L. Zhao, Marketing with ChatGPT: Navigating the ethical terrain of GPT-based chatbot technology 4 (2) 375–384. doi:10.3390/ai4020019.
URL https://www.mdpi.com/2673-2688/4/2/19

[342] D. Kumordzie, All you need to know about ChatGPT & why its a threat to google.
URL https://medium.com/@cdkumordzie/all-you-need-to-know-about-chatgpt-why-its-a-threat-to-google-fd2b887c8ff8

[343] Levente, The pros and cons dark side of using chat GPT for businesses.
URL https://medium.com/@Levente22/the-pros-and-cons-dark-side-of-using-chat-gpt-for-businesses-cf2373119dab

[344] B. Marr, What does ChatGPT really mean for your job?
URL https://www.forbes.com/sites/bernardmarr/2023/02/13/what-does-chatgpt-really-mean-for-your-job/

[345] M. Abdullah, A. Madain, Y. Jararweh, ChatGPT: Fundamentals, applications and social impacts, in: 2022 Ninth International Conference on Social Networks Analysis, Management and Security (SNAMS), pp. 1–8, ISSN: 2831-7343. doi:10.1109/SNAMS58071.2022.10062688.

[346] S. A. Khowaja, P. Khuwaja, K. Dev, ChatGPT needs SPADE (sustainability, PrivAcy, digital divide, and ethics) evaluation: A review. arXiv:2305.03123[cs], doi:10.48550/arXiv.2305.03123.
URL http://arxiv.org/abs/2305.03123

[347] B. Gupta, T. Mufti, S. S. Sohail, D. O. Madsen, ChatGPT: A brief narrative reviewdoi:10.20944/preprints202304.0158.v1.
URL https://www.preprints.org/manuscript/202304.0158/v1

[348] D. Bruff, Teaching in the artificial intelligence age of ChatGPT | teaching + learning lab.
URL https://tll.mit.edu/teaching-in-the-artificial-intelligence-age-of-chatgpt/

[349] A. Iskender, Holy or unholy? interview with open AI's ChatGPT 34 3414–3414. doi:10.54055/ejtr.v34i.3169.
URL https://ejtr.vumk.eu/index.php/about/article/view/3169

[350] M. S. Khan, H. Umer, Chatgpt in finance: Addressing ethical challenges. doi:10.2139/ssrn.4439967.
URL https://papers.ssrn.com/abstract=4439967

[351] H. Lee, The rise of ChatGPT: Exploring its potential in medical education ase.2270doi:10.1002/ase.2270.
URL https://anatomypubs.onlinelibrary.wiley.com/doi/10.1002/ase.2270

[352] G. H. Sun, S. H. Hoelscher, The ChatGPT storm and what faculty can do 48 (3) 119. doi:10.1097/NNE.0000000000001390.
URL https://journals.lww.com/nurseeducatoronline/Fulltext/2023/05000/The_ChatGPT_Storm_and_What_Faculty_Can_Do.1.aspx?context=FeaturedArticles&collectionId=5

[353] A. G. Cardoso, Do we need a chat-GPT-gov? the importance of technology for effective access to public information. doi:10.2139/ssrn.4365773.
URL https://papers.ssrn.com/abstract=4365773

[354] M. Shidiq, The use of artificial intelligence-based chat-gpt and its challenges for the world of education; from the viewpoint of the development of creative writing skills 1 (1) 353–357.
URL https://ejournal.unuja.ac.id/index.php/icesh/article/view/5614

[355] J. Lee, M. Yilmaz Soylu, ChatGPT and assessment in higher education.
URL https://c21u.gatech.edu/sites/default/files/publication/2023/03/C21U%20ChatGPT%20White%20Paper_Final.pdf

[356] D. Mhlanga, Open AI in education, the responsible and ethical use of ChatGPT towards lifelong learning. doi:10.2139/ssrn.4354422.
URL https://papers.ssrn.com/abstract=4354422

[357] A. Bahrini, M. Khamoshifar, H. Abbasimehr, R. J. Riggs, M. Esmaeili, R. M. Majdabadkohne, M. Pasehvar, ChatGPT: Applications, opportunities, and threats. arXiv:2304.09103[cs], doi:10.48550/arXiv.2304.09103.
URL http://arxiv.org/abs/2304.09103

[358] A. M. A. Ausat, B. Massang, M. Efendi, N. Nofirman, Y. Riady, Can chat GPT replace the role of the teacher in the classroom: A fundamental analysis 5 (4) 16100–16106. doi:10.31004/joe.v5i4.2745.
URL https://www.jonedu.org/index.php/joe/article/view/2745

[359] I. Solaiman, M. Brundage, J. Clark, A. Askell, A. Herbert-Voss, J. Wu, A. Radford, G. Krueger, J. W. Kim, S. Kreps, M. McCain, A. Newhouse, J. Blazakis, K. McGuffie, J. Wang, Release strategies and the social impacts of language models. arXiv:1908.

09203[cs], doi:10.48550/arXiv.1908.09203.
URL http://arxiv.org/abs/1908.09203

[360] A. Tiunova, F. Muñoz, Chatgpt: Using ai in social studies academic research. doi:10.2139/ssrn.4451612.
URL https://papers.ssrn.com/abstract=4451612

[361] P. Geertsema, A. Bifet, R. Green, ChatGPT and large language models: What are the implications for policy makers? doi:10.2139/ssrn.4424048.
URL https://papers.ssrn.com/abstract=4424048

[362] A. George, A.S.Hovan George, A.S.Gabrio Martin, The environmental impact of AI: A case study of water consumption by chat GPTdoi:10.5281/ZENODO.7855594.
URL https://zenodo.org/record/7855594

[363] S. S. Biswas, Role of chat gpt in public health, Annals of Biomedical Engineering (2023) 1–2.

[364] S. Biddle, The internet's new favorite AI proposes torturing iranians and surveilling mosques.
URL https://theintercept.com/2022/12/08/openai-chatgpt-ai-bias-ethics/

[365] D. C, P. J, ChatGPT and large language models: what's the risk?
URL https://www.ncsc.gov.uk/blog-post/chatgpt-and-large-language-models-whats-the-risk

[366] E. Now, ChatGPT-4 reinforces sexist stereotypes by stating a girl cannot "handle technicalities and numbers" in engineering.
URL https://www.equalitynow.org/news_and_insights/chatgpt-4-reinforces-sexist-stereotypes/

[367] K. Robson, Do AI chatbots like ChatGPT pose a major cybersecurity risk?
URL https://www.verdict.co.uk/do-ai-chatbots-like-chatgpt-pose-a-major-cybersecurity-risk/

[368] B. Perrigo, Exclusive: The $2 per hour workers who made ChatGPT safer.
URL https://time.com/6247678/openai-chatgpt-kenya-workers/

[369] Z. Li, The dark side of ChatGPT: Legal and ethical challenges from stochastic parrots and hallucination. arXiv:2304.14347[cs], doi:10.48550/arXiv.2304.14347.
URL http://arxiv.org/abs/2304.14347

[370] N. G. Vidhya, D. Devi, N. A, T. Manju, Prognosis of exploration on chat GPT with artificial intelligence ethics 2 (9) 60–69. doi:10.14295/bjs.v2i9.372.

URL https://www.brazilianjournalofscience.com.br/revista/article/view/372

[371] A. Tamkin, M. Brundage, J. Clark, D. Ganguli, Understanding the capabilities, limitations, and societal impact of large language models. arXiv:2102.02503[cs], doi:10.48550/arXiv.2102.02503.
URL http://arxiv.org/abs/2102.02503

[372] C. Treude, H. Hata, She elicits requirements and he tests: Software engineering gender bias in large language models. arXiv:2303.10131[cs], doi:10.48550/arXiv.2303.10131.
URL http://arxiv.org/abs/2303.10131

[373] H. Farrell, A. Newman, J. Wallace, Spirals of delusion how al distorts decision-making and makes dictators more dangerous.
URL https://www.foreignaffairs.com/world/spirals-delusion-artificial-intelligence-decision-making

[374] A. J. Thirunavukarasu, R. Hassan, S. Mahmood, R. Sanghera, K. Barzangi, M. E. Mukashfi, S. Shah, Trialling a large language model (ChatGPT) in general practice with the applied knowledge test: Observational study demonstrating opportunities and limitations in primary care 9 (1) e46599. doi:10.2196/46599.
URL https://mededu.jmir.org/2023/1/e46599

[375] O. Stepanechko, L. Kozub, English teachers' concerns about the ethical use of chat GPT by university students (25) 297–302. doi:10.36074/grail-of-science.17.03.2023.051.
URL https://archive.journal-grail.science/index.php/2710-3056/article/view/1040

[376] N. Helberger, N. Diakopoulos, ChatGPT and the AI act 12 (1). doi:10.14763/2023.1.1682.
URL https://policyreview.info/essay/chatgpt-and-ai-act

[377] C. Air, S. Wijetunge, A. Dimitrov, The ethics of AI: The cyber risks posed by chat GPT.
URL https://www.dacbeachcroft.com/en/gb/articles/2023/february/the-ethics-of-ai-the-cyber-risks-posed-by-chat-gpt

[378] Z. B. Wolf, AI can be racist, sexist and creepy. what should we do about it? | CNN politics.
URL https://www.cnn.com/2023/03/18/politics/ai-chatgpt-racist-what-matters/index.html

[379] C. Vallance, ChatGPT: New AI chatbot has everyone talking to it.
URL https://www.bbc.com/news/technology-63861322

[380] D. Rozado, The political biases of ChatGPT 12 (3) 148. doi:10.3390/socsci12030148.
URL https://www.mdpi.com/2076-0760/12/3/148

[381] I. for Human Rights {and} Business, We asked ChatGPT about its impact on human rights and business. here's what it told us.
URL https://www.ihrb.org/focus-areas/information-communication-technology/we-asked-chatgpt-about-its-impact-on-human-rights-on-business-heres-what-it-told-us

[382] S. Biswas, Prospective role of chat GPT in the military: According to ChatGPTdoi:10.32388/8WYYOD.
URL https://www.qeios.com/read/8WYYOD

[383] Z. N. Khlaif, Ethical concerns about using AI-generated text in scientific research. doi:10.2139/ssrn.4387984.
URL https://papers.ssrn.com/abstract=4387984

[384] G. per la protezione dei dati personali, ChatGPT: OpenAI riapre la piattaforma in italia garantendo più trasparenza e più diritti a utenti e non utenti europei.
URL https://www.garanteprivacy.it:443/home/docweb/-/docweb-display/docweb/9881490

[385] N. Curtis, ChatGPT§, To ChatGPT or not to ChatGPT? the impact of artificial intelligence on academic publishing 42 (4) 275. doi:10.1097/INF.0000000000003852.
URL https://journals.lww.com/pidj/Citation/2023/04000/To_ChatGPT_or_not_to_ChatGPT__The_Impact_of.1.aspx

[386] A. Gabbiadini, O. Dimitri, C. Baldissarri, A. Manfredi, Does ChatGPT pose a threat to human identity? doi:10.2139/ssrn.4377900.
URL https://papers.ssrn.com/abstract=4377900

[387] T. Telegraph, Microsoft, OpenAI, alphabet and big tech are ignoring the human cost behind the rise of ChatGPT and other AI-powered chatbots.
URL https://techtelegraph.co.uk/microsoft-openai-alphabet-and-big-tech-are-ignoring-the-human-cost-behind-the-rise-of-chatgpt-and-other-ai-powered-chatbots/

[388] C. DiBenedetto, ChatGPT's surprisingly human voice came with a human cost.
URL https://mashable.com/article/chat-gpt-open-ai-workers-exploitation

[389] A. Al Ashry, Chat GPT and its legal impact on society as a new form of AI - copyright - united arab emirates.
URL https://www.mondaq.com/copyright/1299882/chat-gpt-and-its-legal-impact-on-society-as-a-new-form-of-ai

[390] J. Bareis, We are scared of the question chat-GPT cannot answer. because the answer is too obvious. doi:10.2139/ssrn.4410324.
URL https://papers.ssrn.com/abstract=4410324

[391] J. Rutinowski, S. Franke, J. Endendyk, I. Dormuth, M. Pauly, The self-perception and political biases of ChatGPT. arXiv:2304.07333[cs], doi:10.48550/arXiv.2304.07333.
URL http://arxiv.org/abs/2304.07333

[392] F. Suguri Motoki, V. Pinho Neto, V. Rodrigues, More human than human: Measuring ChatGPT political biasdoi:10.2139/ssrn.4372349.
URL https://ueaeprints.uea.ac.uk/id/eprint/91668/

[393] W. Solution, What is ChatGPT and the benefits of using ChatGPT.
URL https://www.weetechsolution.com/blog/what-is-chat-gpt-and-the-advantages-of-using-chat-gpt

[394] T. Y. Zhuo, Y. Huang, C. Chen, Z. Xing, Exploring AI ethics of ChatGPT: A diagnostic analysis. arXiv:2301.12867[cs], doi:10.48550/arXiv.2301.12867.
URL http://arxiv.org/abs/2301.12867

[395] A. Sobieszek, T. Price, Playing games with ais: The limits of GPT-3 and similar large language models 32 (2) 341–364. doi:10.1007/s11023-022-09602-0.
URL https://doi.org/10.1007/s11023-022-09602-0

[396] Y. Cao, S. Li, Y. Liu, Z. Yan, Y. Dai, P. S. Yu, L. Sun, A comprehensive survey of AI-generated content (AIGC): A history of generative AI from GAN to ChatGPTdoi:10.48550/ARXIV.2303.04226.
URL https://arxiv.org/abs/2303.04226

[397] R. Rodrigues, Legal and human rights issues of ai: Gaps, challenges and vulnerabilities, Journal of Responsible Technology 4 (2020) 100005. doi:10.1016/j.jrt.2020.100005.

[398] M. T. Baldassarre, D. Caivano, B. Fernandez Nieto, D. Gigante, A. Ragone, The social impact of generative ai: An analysis on chatgpt, in: Proceedings of the 2023 ACM Conference on Information Technology for Social Good, CM Conference on Information Technology for Social Good, 2023, pp. 363–373. doi:10.1145/3582515.3609555.

[399] D. Milmo, Chatgpt reaches 100 million users two months after launch, The Guardianhttps://www.theguardian.com/technology/2023/feb/02/chatgpt-100-million-users-open-ai-fastest-growing-app (February 2 2023).

[400] V. Türk, Artificial intelligence must be grounded in human rights, says high commissioner (2023).

URL https://www.ohchr.org/en/statements/2023/07/artificial-intelligence-must-be-grounded-human-rights-says-high-commissioner

[401] United Nations, Peace, dignity and equality on a healthy planet, https://www.un.org/en/.
URL https://www.un.org/en/

[402] Amnesty International, Amnesty international, https://www.amnesty.org/en/ (2023).
URL https://www.amnesty.org/en/

[403] European Center for Not-for-Profit Law Stichting, Home, https://ecnl.org/ (2023).

[404] Access Now, Artificial intelligence, https://www.accessnow.org/issue/artificial-intelligence/ (2023).
URL https://www.accessnow.org/issue/artificial-intelligence/

[405] AI Now Institute, Home, https://ainowinstitute.org/ (2023).
URL https://ainowinstitute.org/

[406] I. Gabriel, M. Mitchell, T. Gebru, V. Prabhakaran, A human rights approach to responsible ai (2022). arXiv:2210.02667.
URL https://arxiv.org/pdf/2210.02667.pdf

[407] Legal and human rights issues of AI: Gaps, challenges and vulnerabilities, Journal of Responsible Technology 4 (2020) 100005. doi:10.1016/j.jrt.2020.100005.

[408] F. Kamei, I. Wiese, C. Lima, I. Polato, V. Nepomuceno, W. Ferreira, M. Ribeiro, C. Pena, B. Cartaxo, G. Pinto, S. Soares, Grey literature in software engineering: A critical review, Information and Software Technology 138 (2021) 106609. doi:10.1016/j.infsof.2021.106609.
URL https://doi.org/10.1016/j.infsof.2021.106609

[409] C. Rikap, Same end by different means: Google, amazon, microsoft and facebook's strategies to dominate artificial intelligence, SSRN Scholarly Paper (4472222) (2023). doi:10.2139/ssrn.4472222.
URL https://doi.org/10.2139/ssrn.4472222

[410] OpenAI, Openai (2023).
URL https://openai.com/

[411] Meta AI, About meta ai, https://ai.meta.com/about/ (2023).

[412] Google AI, Research, https://ai.google/discover/research/ (n.d.).

[413] Microsoft, Microsoft AI, https://news.microsoft.com/ai/ (2023).

[414] M. T. Baldassarre, D. Caivano, D. Gigante, B. F. Nieto, A. Ragone, Online appendix, https://figshare.com/s/57ef5620007720ce5657 (2023).
URL https://figshare.com/s/57ef5620007720ce5657

[415] European Commission, Ethics guidelines for trustworthy ai (2019).
URL https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai

[416] S. Barocas, A. D. Selbst, Big data's disparate impact, California Law Review 104 (2016) 671.
URL https://api.semanticscholar.org/CorpusID:143133374

[417] J. I. Garcia-Gathright, A. Springer, H. Cramer, Assessing and addressing algorithmic bias - but before we get there, ArXiv abs/1809.03332 (2018).
URL https://api.semanticscholar.org/CorpusID:52181973

[418] M. Veale, M. Van Kleek, R. Binns, Fairness and accountability design needs for algorithmic support in high-stakes public sector decision-making, in: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI '18, Association for Computing Machinery, New York, NY, USA, 2018, p. 1–14. doi:10.1145/3173574.3174014.
URL https://doi.org/10.1145/3173574.3174014

[419] D. Fallis, What is disinformation?, Library Trends 63 (3) (2015) 401–426. doi:10.1353/lib.2015.0014.

[420] W. Warner, J. Hirschberg, Detecting hate speech on the world wide web, in: Proceedings of the Second Workshop on Language in Social Media, Association for Computational Linguistics, Montréal, Canada, 2012, pp. 19–26.
URL https://aclanthology.org/W12-2103

[421] J. T. Nockleby, Hate speech, in: L. W. Levy, K. L. e. a. Karst (Eds.), Encyclopedia of the American Constitution, 2nd Edition, Macmillan, 2000, pp. 1277–1279.
URL http://www.jiffynotes.com/a_study_guides/book_notes/eamc_03/eamc_03_01193.html

[422] J. Curzon, T. A. Kosa, R. Akalu, K. El-Khatib, Privacy and artificial intelligence, IEEE Transactions on Artificial Intelligence 2 (2) (2021) 96–108. doi:10.1109/TAI.2021.3088084.

[423] H. F. Nissenbaum, Privacy in Context: Technology, Policy, and the Integrity of Social Life, Stanford Law Books, 2010.

[424] ENISA, Definition of cybersecurity - gaps and overlaps in standardisation, https://www.enisa.europa.eu/publications/definition-of-cybersecurity (2016-07-01).

[425] I. E. Koch, Human Rights as Indivisible Rights: The Protection of Socio-economic Demands under the European Convention on Human Rights, Martinus Nijhoff Publishers, 2009.