TURUN
YLIOPISTO
UNIVERSITY
OF TURKU

Data protection software engineering techniques

Practical research into the demands of the GDPR

Kalle Hjerppe

TURUN YLIOPISTON JULKAISUJA – ANNALES UNIVERSITATIS TURKUENSIS
SARJA – SER. F OSA – TOM. 79 | TECHNICA – INFORMATICA | TURKU 2026

TURUN
YLIOPISTO
UNIVERSITY
OF TURKU

DATA PROTECTION SOFTWARE ENGINEERING TECHNIQUES

Practical research into the demands of the GDPR

Kalle Hjerppe

TURUN YLIOPISTON JULKAISUJA ANNALES UNIVERSITATIS TURKUENSIS
SARJA SER. F OSA TOM. 79 | TECHNICA INFORMATICA | TURKU 2026

University of Turku

Faculty of Technology
Department of Computing
Information and Communication Technology
Doctoral Programme in Technology

Supervised by

Prof. Ville Leppänen, Ph.D.
University of Turku
Finland

Asst. Prof. Jukka Ruohonen, D.Sc. (Tech.)
University of Southern Denmark
Denmark

Adj. Prof. Johannes Holvitie, D.Sc. (Tech.)
University of Turku
Finland

Reviewed by

Prof. Daniel Mendez Fernandez, Ph.D.
Blekinge Institute of Technology
Sweden

Assoc. Prof. Jose M. Del Alamo, Ph.D.
Universidad Politécnica de Madrid
Spain

Opponent

Prof. Tomi Männistö, D.Sc. (Tech.)
University of Helsinki
Finland

ISBN 978-952-02-0592-8 (PRINT)
ISBN 978-952-02-0593-5 (PDF)
ISSN 2736-9390 (PRINT)
ISSN 2736-9684 (ONLINE)
Painosalama, Turku, Finland, 2026

Vapaan tieteen lahja
vapaalle miehelle

UNIVERSITY OF TURKU
Faculty of Technology
Department of Computing
Information and Communication Technology
HJERPPE, KALLE: Data protection software engineering techniques
Doctoral dissertation, 165 pp.
Doctoral Programme in Technology
April 2026

ABSTRACT

Personal data is processed frequently and for important and not-so-important purposes in the connected software systems of the internet age. The European Union (EU) has recognized the importance of personal data and commits to protecting the fundamental rights to privacy and to data protection. Data protection can be summarized as the concept that requires those processing personal data to process it lawfully, fairly, and transparently. The General Data Protection Regulation (GDPR) is the main source of the concrete personal data processing rules in the EU. Among other things, the rights provided in the GDPR have implications to software systems and their development. This dissertation studies the nuances of these requirements and their implementation, in order to improve the understanding of the regulation and its implications, and develops novel software engineering techniques, in order to improve the state-of-the-art of data protection engineering. The exploratory research approach employs a variety of methods across five independent publications with qualitative and quantitative elements, and provides design science contributions. The four research questions explore the technical requirements of the GDPR, how meeting of its requirements can be improved, specifically via static analysis of software source code, and how the different industry stakeholders align on data protection. The concrete contributions include requirements engineering analysis, static analysis methods for personal data flows and composable privacy policies, an analysis of GDPR enforcement actions, and thematic analysis of device sharing data protection strategies. This dissertation claims that (a) software engineering as an art ought to raise the standard of data protection in an interdisciplinary undertaking, (b) there are improvements available in software architecture, static analysis, and ecosystem collaboration, (c) information about personal data processing can be embedded into software at the source code level with a reasonable effort, which ought to be considered as the GDPR "state-of-the-art" protection measures, and (d) the results altogether can be viewed as an actionable road map for improved data protection across software engineering in general.

KEYWORDS: Software engineering, data protection, GDPR, static analysis, privacy

TURUN YLIOPISTO
Teknillinen tiedekunta
Tietotekniikan laitos
Tietotekniikka
HJERPPE, KALLE: Data protection software engineering techniques
Väitöskirja, 165 s.
Teknologian tohtoriohjelma
Huhtikuu 2026

TIIVISTELMÄ

Henkilötiedon käsittely on yleistä sekä tärkeisiin että vähemmän tärkeisiin tarkoituksiin internetin aikakauden yhdistetyissä ohjelmistoista koostuvissa järjestelmissä. Euroopan unioni (EU) on tunnistanut henkilötiedon käsittelyn merkityksen ja on sitoutunut yksityisyyden ja tietosuojan perusoikeuksiin. Tietosuojan voi konseptina tiivistää vaatimukseksi siitä, että henkilötietoa käsittelevät tekevät sitä lainmukaisesti, reilusti ja läpinäkyvästi. Yleinen tietosuoja-asetus (GDPR) on tärkein lähde konkreettisille henkilötiedon käsittelyn säännöille EU:ssa. Asetuksen suomilla henkilötiedon suojan oikeuksilla on vaikutuksia ohjelmistoille ja niiden kehittämiselle. Väitöskirjassa esitetään tutkimus, jossa tietosuojavaatimusten yksityiskohtia ja toteuttamista arvioidaan, tuottaakseen ymmärrystä asetuksesta ja sen vaikutuksista, sekä kehittääkseen uusia ohjelmistotuotannon menetelmiä tietosuojan parantamiseksi. Työn kartoittava ja kokeileva tutkimus käyttää useaa menetelmää, koostuen viidestä itsenäisestä julkaisusta ja sisältäen sekä kvalitatiivista että kvantitatiivista tutkimusta. Väitöskirjan neljä tutkimuskysymystä valottavat GDPR:n teknisiä vaatimuksia, miten vaatimustenmukaisuutta voi helpottaa (erityisesti staattisen analyysin keinoin), ja miten eri ohjelmistoteollisuuden osalliset ja toimijat osaltaan tietosuojaan koskevat. Väitöskirjan konkreettiset kontribuutiot ovat mm. vaatimusmäärittelyä, staattisen analyysin menetelmiä henkilötiedon kululle ja tietosuojaselosteen sisällölle, analyysin viranomaisten seuraamustoimenpiteistä, sekä temaattisen analyysin tietosuojakeinoista jaettujen laitteiden käyttötapauksissa. Tuloksien perusteella väitän, että (a) ohjelmistotuotannon tulisi nostaa tietosuojan tasoa monialaisella tavalla, (b) edellä mainittuun tason nostoon on mahdollisuuksia ohjelmistoarkkitehtuurissa, staattisessa analyysissä ja ekosysteemin yhteistyössä, (c) tietoa henkilötiedon käsittelystä voidaan upottaa ohjelmistojen lähdekoodiin kohtuullisella työllä, mikä tulisi ottaa huomioon GDPR:n “uusimpana tekniikkana”, ja että (d) väitöskirjan tuloksia voi kokonaisuutena tarkastellen pitää toimenpidekelpoisena tiekarttana tietosuojamenetelmien parantamiseksi ohjelmistotuotannossa yleisesti.
ASIASANAT: Ohjelmistosuunnittelu, tietosuoja, GDPR, staattinen analyysi, yksityisyys

Acknowledgements

Here it is, the end of this chapter of the journey is in sight. Here, as I put this final touch to the manuscript, I am awed with gratitude. It has been a great pleasure to pursue doctoral studies and research in the University of Turku. Looking back to the beginning, it was a different man, from a different world, who began the work.

What a privilege it is to have been supervised and guided by such esteemed experts and kind persons as Ville Leppänen, Jukka Ruohonen, and Johannes Holvitie. Without your warm encouragement and collaboration every step of the way, I would not have started – or finished – this dissertation path. Ville, you provide the perfect blend of guidance and freedom. Jukka, you are an exemplar in substance, and I am honoured to do research with you. Johannes, you have supported me through thick and thin. Thank you. Furthermore, I am grateful to Professors Mendez and Del Alamo for their insightful and constructive pre-examination of this dissertation, and to Professor Tomi Männistö for agreeing to act as my opponent.

During this and related research, I have had the pleasure of collaborating with many bright and resourceful colleagues in an environment that radiates with academic ideals. Sampsa Rauti, Sini Mickelsson, Juha Vesala, Timi Heino, Robin Carlsson, Kalle Rindell, Katleena Kortesuo, Eun-Young Kang, Maximilian Von Zastrow, Sammani Rajapaksha, among others, and everyone at the department of computing: thank you all. Furthermore, during the doctoral studies, I have been fortunate to have been employed by very supportive employers, namely Geniem, VSSHP, Varha, and the University of Turku in both the faculties of computing and law. In each organization, I have found a community, and I am grateful to every one of you.

In my life, my motivation and well-being are sustained by my dear friends and extended family, a fact for which I am deeply grateful. Your support and company are everything. KS. Too many names to enumerate, but I especially wish to acknowledge my mother, Päivi, and my late father, Markus. Finally, Tuuli, my love, you have my deepest gratitude – none of this would have been written without you.

28th February 2026
Kalle Hjerppe

Table of Contents

List of Original Publications

This dissertation is based on the following five original publications, which have been reproduced with the permission of the copyright holders:

$\mathcal{P}_1$ Hjerppe, K., Ruohonen, J., Leppänen, V. (2019). The General Data Protection Regulation: Requirements, Architectures, and Constraints. *2019 IEEE 27th International Requirements Engineering Conference (RE 2019).* (pp. 265-275.) IEEE.

$\mathcal{P}_2$ Hjerppe, K., Ruohonen, J., Leppänen, V. (2020). Annotation-Based Static Analysis for Personal Data Protection. *In: Friedewald, M. et al. (Eds.) Privacy and Identity Management. Data for Better Living: AI and Privacy. Privacy and Identity 2019. 14th IFIP WG 9.2, 9.6/11.7, 11.6/SIG 9.2.2 International Summer School, Windisch, 2019, Revised Selected Papers. IFIP Advances in Information and Communication Technology*, (vol. 576: pp. 343–358.) Springer.

$\mathcal{P}_3$ Hjerppe, K., Ruohonen, J., Leppänen, V. (2022). Extracting LPL privacy policy purposes from annotated web service source code. *Software and Systems Modeling*, (vol. 22: pp. 331–349.) Springer.

$\mathcal{P}_4$ Ruohonen, J., Hjerppe, K. (2022). The GDPR enforcement fines at glance. *Information Systems*, (vol 106: art. 101876.) Elsevier.

$\mathcal{P}_5$ Hjerppe, K., Mickelsson, S., Ruohonen, J., Carlsson, R., Heino, T., Rauti, S., & Leppänen, V. (2025). Device sharing features: a study on software policy approaches and platform capabilities. International Review of Law, Computers & Technology, (1–38.) Taylor & Francis.

During the course of doctoral studies, the following peer-reviewed publications, related but not included in the dissertation, were produced:

$\mathcal{P}_6$ Ruohonen, J., Hjerppe, K., & Rindell, K. (2021). A Large-Scale Security-Oriented Static Analysis of Python Packages in PyPI. *2021 18th International Conference on Privacy, Security and Trust (PST)* (pp. 1-10). IEEE.

$\mathcal{P}_7$ Ruohonen, J., Hjerppe, K., and von Zastrow, M. (2024). An Exploratory Case Study on Data Breach Journalism. *In Proceedings of the 19th International Conference on Availability, Reliability and Security (ARES '24).* (art. 140, pp. 1–9). ACM.

$\mathcal{P}_8$ Ruohonen, J., Hjerppe, K. (2025). The Potential of Citizen Platforms for Requirements Engineering of Large Socio-Technical Software Systems. *In Requirements Engineering: Foundation for Software Quality (REFSQ 2025).* LNCS, (vol 15588.) Springer.

$\mathcal{P}_9$ Ruohonen, J., Hjerppe, K., Kang, E.-Y. (2025). A Mapping Analysis of Requirements Between the CRA and the GDPR. *In Proceedings of the 2025 IEEE 33rd International Requirements Engineering Conference Workshops. The 12th International Workshop on Evolving Security & Privacy Requirements Engineering (ESPRE 2025).* IEEE.

1 Introduction

This dissertation work originates from the simple curiosity of a software development practitioner: how can we make better software in a better way? In the agile world of *DevSecOps* [1] et cetera, an increasing amount of responsibility is put on the shoulders of interdisciplinary teams with relatively few members. This is efficient. However, the skills and breadth of knowledge required of the engineers also increases. These *'enlightened engineers'* must, as a team, possess skills of specialized topics such as security, data protection and privacy, or usability and accessibility. (Might as well do research, too.) The topic and focus of this dissertation is on data protection and privacy, from the perspectives of software and system architectures, and requirements engineering. Privacy can be more than just a compliance checkbox item. These are important domains of software engineering to improve globally, and the fundamental rights of all of us are at stake. On a smaller scale, this dissertation serves as a demonstration of learning and contributing to the academic knowledge of new domain as an industry practitioner.

Software engineering is the discipline of developing and maintaining software in a systematic way. The sum of the knowledge of the entire domain is known as *the Software Engineering Body of Knowledge*, or SWEBOK, which can be structured into knowledge areas that cover the whole life cycle of software [2]. This dissertation builds upon this body and applies knowledge from the SWEBOK knowledge areas of requirements, design, modeling, processes, and economics. Software and system architecture is the structure and organization of components, their interfaces, and the trade-offs between options [2]. Architectures can be grouped in categories that follow similar principles, such as service-oriented architectures (SOAs), or architectures of a particular perspective, such as integration architectures. While software in general is easily modifiable, even after deployment, compared to the traditional engineering disciplines, architectural decisions are commonly those with the most weight and commitment required. It is generally accepted that system-wide quality attributes, such as security and privacy, are easier to build into software rather than patch in later [2]. The architectural decisions for these therefore warrant special consideration.

Data protection and privacy are fundamental rights of the data subjects, the natural persons whose data is being processed by our information systems [3]. Privacy is an ephemeral concept, which cannot, and perhaps ought not to, be rigidly defined. It

can concern, for example, goals and means, and it is contextual and interdependent. Solove has proposed six overlapping, non-taxonomical categories of family resemblances: the right to be left alone, limited access to self, secrecy and concealment, control over personal information, personhood – protection of identity and dignity, and intimacy [3; 4]. Data protection, on the other hand, is a closely related concept that originates from the right to the protection of personal data. It has overlap with privacy, but also requires that personal data is processed fairly, for a specified purpose, and with a legitimate basis [3]. Information security is another related term that covers the methods to ensure resource confidentiality, integrity and availability – pre-requisites for privacy or data protection. Within software engineering, the goals *privacy by design* is corresponded by privacy engineering and *data protection by design* by data protection engineering are domains that concern themselves with creating methods to tightly integrate the rights of the data subjects to the structure of software and systems, and to the software development life cycle, with technological mechanisms as opposed to compliance by policy only [5]. This dissertation is focused on data protection; the methods how an organization that develops systems that process personal data can honor the rights of the data subjects with less cost, less complexity, and reduced risks. A secondary focus is on ordinary software, with the goal of developing methods that can apply to legacy code bases.

Before technical measures, in order to design, software architects must have an understanding of the *requirements* of the system in question. This process can be iterative and correct its course on the way, to some extent, but the aforementioned concern of architectural decisions still applies. When it comes to regulatory requirements, such as data protection, at a minimum what must be judged is whether the system is compliant or not. The issue is not simple, beyond trivial cases, as there are a myriad of factors and stakeholders to consider, and just interpreting the law is an endless task [6]. At some point in the software design process, we must bridge the gap from interpretative law into explicit program instructions. The theoretical framework for viewing the issue in this dissertation is modeled on two society-level activities: *the regulatory technology transfer activity* [7; 8] and *the software service-usage-enforcement feedback activity* [6]. These are elaborated on in later chapters. The present dissertation contributes materials to the building of this inter-disciplinary bridge, with the motivation that we, as societies, do not definitely know how to regulate software nor how to comply with regulation. Academic knowledge is used in policy preparation, but the actual magnitude of evidence-based policy making is not within the scope of this dissertation.

As Wohlin *et al.* put it into words, a software engineering organization must continuously learn and improve its understanding of the software process and product, in order to succeed in the software business [8]. The same thought can be extended into the entire ecosystem, including technology transfer from the academia into industry. Empirical studies are a tool to facilitate this process, as a way to evaluate changes be-

fore committing the entire organization behind them [8]. The Quality Improvement Paradigm, as described by Vasili already in 1985 [9; 8] is, in essence, a prototypical agile iterative methodology, with two distinct features: the Experience Factory to organize institutional knowledge, and the Goal-Question-Metric (GQM) model to tie empiricism into project goals and structure objectives on the appropriate abstraction level. This GQM framework is the conceptual structure the present dissertation work adapts to organize its approach.

The GQM is a hierarchy of three abstraction levels: the conceptual level, the operational level, and the metric level [10; 8]. The objects on the conceptual level are *goals*, which are defined as purposes regarding an issue within a domain, from a certain viewpoint. The operational level defines questions which characterize the quality of the objects of the goals. Finally, the questions are, in turn, answered by quantitative metrics that are interpreted from the perspectives of the questions.

Within this framework, on the conceptual level, the dissertation research has the following two goals:

$\mathcal{G}_1$ *Improve the understanding of the GDPR and its implications for software design, from the viewpoint of software engineering practitioners.*

$\mathcal{G}_2$ *Improve data protection methods for software and system architectures and development processes, from the viewpoint of software engineering practitioners.*

On the operational level, the dissertation considers the following set of dissertation's research questions *($\mathcal{D}.\mathcal{RQ}$)*:

$\mathcal{D}.\mathcal{RQ}_1$ *What does the GDPR require of software-as-a-service technologies and their providers, and what are the implications of these requirements?*

$\mathcal{D}.\mathcal{RQ}_2$ *How can meeting GDPR requirements in software be improved from the perspectives of regulators, end-users, and in terms of software architecture patterns?*

$\mathcal{D}.\mathcal{RQ}_3$ *In what ways can information about personal data processing be represented within programming language type systems, and how can such representations be applied in software engineering practice?*

$\mathcal{D}.\mathcal{RQ}_4$ *How do data protection requirements align between business objectives, privacy policies, and systems and software architecture?*

This work makes inroads towards the questions with the elephant-eating method: one bite at a time. The research questions are on an abstract level, and their answers

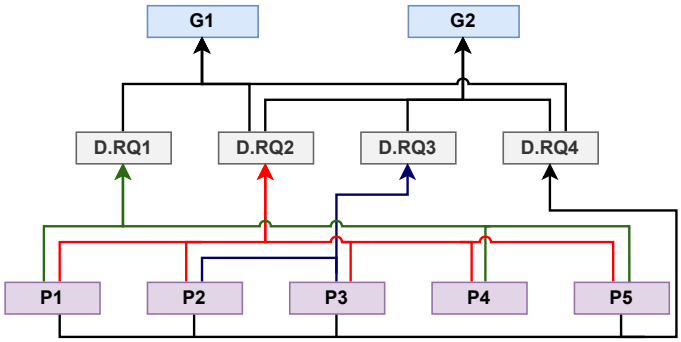

Figure 1. Overview of the Goal-Question-Metric -mapping of research questions into publications containing research metrics.

are composed of a set of concrete contributions (metrics). To define more concrete work packages for these topics, the questions are further refined into and in the individual original publications $\mathcal{P}_1$–$\mathcal{P}_5$, with their own granular publications' research questions *($\mathcal{P}.\mathcal{RQ}s$)*. These represent the quantitative level in the GQM framework, and each contribute in part to the answering of the operational level questions. The questions are approached with an array of research methods: theoretical research, literature review, grounded theory, content analysis, case study, descriptive statistics, and exploratory testing. The original research is further described in Chapter 3, but a summary of this approach with mapping the publications' contributions to the research questions is illustrated in Figure 1.

The main results of the research work include, among other lesser contributions:

1. a model of the GDPR requirements for software architecture and patterns to meet them in a limited context;

2. a static-analysis method to model personal data flows in software;

3. a theory and a method to integrate software composition into privacy policy purposes;

4. an analysis of GDPR enforcement actions; and

5. a synthesis of industry strategies for handling the data protection issues in device sharing.

To distill the contributions into summarized takeaways, the dissertation has reached and defends the following claims (the further context and justification of which are elaborated in Chapter 3):

$\mathcal{C}_1$ The GDPR requires organizations to raise the baseline standard of data protection in industry software-as-a-service systems, through an undertaking that is inherently interdisciplinary. Although the regulation can be translated into technical requirements with relative clarity, compliance extends beyond information security. Regulators place emphasis on adherence to the GDPR's fundamental principles, including lawfulness of processing and accountability. This ought to be considered in software engineering processes.

$\mathcal{C}_2$ The current industry practices for software engineering can be improved, in particular the state-of-the-art of the technical measures for data protection required by the GDPR. These improvements are available and reachable with reasonable efforts, as demonstrated by the artifacts designed for particular cases as a part of this dissertation. Potential avenues for improvement include software architectural techniques, static analysis methods, and ecosystem collaboration. Data protection issues are not simple, and require careful consideration in order to reach solutions that satisfy complex user and data controller needs.

$\mathcal{C}_3$ It is reasonably practical to embed information about personal data processing into the semantics of traditional software. This can be used to document processing or to reduce accidental errors, among other use cases. Static analysis tools are within state-of-the-art technical measures for data protection, and the capability to make use of them ought to be considered in design decisions for software architecture. Semantics for personal data processing can be controlled in ordinary software without major architecture overhauls such as resorting to academic programming languages. In fast and agile software development, up-to-date information of how personal data is processed is present in the software source code. These traits ought to be made use of.

$\mathcal{C}_4$ The following playbook is a path with potential to improved data protection in ordinary transaction processing software, where the intermediary steps each independently provide practical value and can be completed without major industry co-operation:

Step 1: Software architects ought to ensure the ability to translate legal requirements into actionable software requirements thereby bridging the gap between legal frameworks and system design.

Step 2: Architect software in a way that enables reasoning about the network of intra and inter system dependencies, for example SOA and static analysis facilities.

Step 3: Apply static analysis to gain a complete model of personal data processing semantics in the software stack.

Step 4: Apply the above model in privacy policies and service descriptions to empower data subjects and ecosystem actors.

Step 5: Apply the above in industry collaboration to develop common data models and personal data features across platform providers and users.

Beyond the current chapter, this dissertation is structured into three chapters which present the synthesis of the original publications following them. The publications, in turn, contain the main research contributions. Chapter 2 provides an overview of the theoretical background and the academic body of knowledge this work builds on. Chapter 3 further describes the research design, methods, the original publications and their results, and the above claims and their justification based on the research results. Chapter 4 concludes the dissertation by outlining the limitations of the research, and discussing its implications and avenues for future work.

2 Background

Before going further into the original research undertaken in this dissertation, it is appropriate and expected to define the topics and to give an overview of the scholarly work this dissertation builds upon. This chapter provides a high-level picture of the main academic disciplines the intersection of which this work sits at. As per the title of the dissertation, *Data Protection Software Engineering Techniques*, the main categories of data protection and software engineering are evident. These are respectively the two main sections this chapter is organized into, with subsections to elaborate on the most relevant sub-disciplines.

2.1 Data protection and privacy

This section is elucidates the background of the disciplines data protection and privacy. The section is arranged into two subsections: the first outlines the concepts and the regulation behind the requirements, and the second the organization governance persepctive including privacy policies.

2.1.1 Concepts and regulation

Data protection and privacy are two separate but closely related concepts, and the meanings of the terms are overloaded and context-dependent. They can be seen among other views as philosophical *notions*, human rights and needs, categories of law, scholarly disciplines, technical terms, or just titles on a compliance checklist. To gain a grasp on the topic of this dissertation, it is best to start from the beginning: data protection and privacy as concepts. Following that, this section elaborates how data protection and privacy are regulated and their governance in an organization that processes personal data. Much of the structure of the following treatment of the concepts and their regulatory history is inspired by, and moreover frequently refers to, the excellent textbook of Hildebrandt [3].

To set the appropriate scope for reaching the goals of this dissertation, we must be mindful to use wide enough definitions of the concepts of privacy and data protection. Privacy will be defined first, as the more foundational concept, and data protection shall follow. Information security research often takes a technical view of privacy as a subset of security, where maintaining the CIA-triad of confidentiality, integrity, and

availability is sufficient to fulfill privacy [3]. These technical measures are integral for achieving privacy, but privacy as a *human right* goes further than this, and must consider the full context of the situation of the data processing.

As a concept, privacy is elusive. In the last fifty or so years, this has caused difficulty for legal scholars, as fundamental regulation protects privacy and the overall landscape has evolved considerably in the information age. It might very well be impossible to truly define the notion exhaustively. While raising this concern [4], even identifying it as essential issue for freedom and democracy, Daniel Solove, thankfully, provides us the intellectual tools to make some sense: Wittgenstein's *family resemblances*. This notion recognizes that there is no 'core' to privacy, rather its nature is dynamic and contextual. Instead of trying to create a taxonomy, it presents privacy as a set of related, overlapping categories: the right to be left alone, limited access to self, secrecy and concealment, control over personal information, personhood – protection of identity and dignity, and intimacy [4; 3]. These are both goals and means, and interdependent. In terms of freedoms, the right to privacy can be formulated as containing both a negative freedom, "the right that others refrain from interference", and a positive freedom, "the freedom to determine how personal information is shared and used" [3].

After this laboured definition, to reiterate, for the purposes of this dissertation, the term *privacy* refers to this wide understanding of the notion. Exact semantics of the word are not important for this dissertation, but this widening of scope is crucial. Privacy protection in the context of an information system cannot merely consist of a set of technical conditions.

Let us then define *data protection* and elaborate on how privacy and data protection relate to each other. Data protection is, alongside privacy, a fundamental right in the European Union (EU) – among other jurisdictions, but EU is the territorial focus of this dissertation. The Charter of Fundamental Rights of the European Union (CFREU) [11] defines the right to privacy (respect for private and family life) and the right to protection of personal data, in its articles 7 and 8, respectively. The article 8 mandates that *"everyone has the right to the protection of personal data concerning him or her"* and that *"such data must be processed fairly for specified purposes and on the basis of the concent of the person concerned or some other legitimate basis laid down by law. Everyone has the right of access to data which has been collected concerning him or her, and the right to have it rectified."* This right is a relatively new development in a historical context, as the charter has been in force only since 2009, and no other constitution or human rights treaty attributes this right to personal data protection [3]. Being enacted to the status of a fundamental right as opposed to a common one is the result of a long development and a weighty outcome [12].

The rights to privacy and data protection can be viewed as a pair of opacity and transparency rights. The former provides a bubble of opacity, as a negative right to prevent others from interfering, and the latter a positive right that when personal

data *is* processed, it must be done in a fair and lawful manner [3]. The principles of data protection are lawfulness, fairness and transparency, purpose limitation, data minimisation, accuracy, storage limitation, integrity and confidentiality, and accountability [13]. The two rights, with some overlap, complement each other – the data protection requirements apply even in cases where privacy is waived. Other aspects of privacy, on the other hand, go beyond personal data. For the purposes of this dissertation, the right to data protection results in obligations to organizations (and anyone other than natural persons for their private purposes) that process personal data in their operations, and is therefore especially relevant to the design of software systems. To summarize the definition of data protection in the present work, it refers to protecting the fundamental rights and freedoms of data subjects when processing personal data.

As it was alluded to previously, and apparent in the subtitle *Practical research into the demands of the GDPR*, this dissertation is focused into the EU regulatory landscape in data processing. The primary context and target of investigation is the General Data Protection Regulation [13], Regulation (EU) 2016/679, which came into force in 2018. The GDPR is, in the end, merely a step in the long path of the development of privacy and data protection in Europe, with many adjacent regulations and directives as well. Some exposition to the history of privacy and data protection regulation is warranted, and follows.

The concept of human rights is a result of moving away from feudalism towards rule of law. The English (1689), French (1789) and US (1791) regulations are the first occurrences of human rights in law, but it took until 1948 for the concepts to reach international law, in the wake of the world wars, via the *Universal Declaration of Human Rights* [3]. These rights were first formulated to defend a citizen against the state, but the need has evolved to protect humans against other powerful societal forces, as exemplified by the data protection rights. The right to privacy has changed in its essence from the distinction of public matters and those of families in Rome to the individualistic modern views of privacy [14]. Constitutional law is the main protection for privacy in many nation states, and for instance, the right to privacy has been 'read into' the US Bill of Rights by interpretation [3]. The United Nations has since provided protection for privacy in international law, with the *International Covenant on Civil and Political Rights* of 1966 [15]. In the EU, this is further reinforced by CFREU, as mentioned.

The origins of data protection law date to the 1970s, presumably motivated by, or at least coinciding with, the development and proliferation of information systems. The principles of data protection, enumerated previously, have remained more or less the same in essence since the 1980 "Fair Information Principles" published by the OECD [3]. Data protection regulation has at first appeared on a national level in the EU, and, in 1975, the European Parliament adopted a resolution appointing the member states the responsibility for data protection [16]. To unify the treatment

across the EU, in order to improve the single market, a common Data Protection Directive (DPD, 95/46/EC) was adopted [16] in 1995. It was later deemed that the digital environment had outgrown the DPD, and that the directive did not provide enough harmonization nor the enforcement effect aspired for [17]. Thus, in order to meet these concerns, the GDPR was developed. The GDPR did not entirely upend the rules of the DPD, although it resulted in a long list of business requirements [18], but its status as a regulation rather than a directive provides it with the supranational effect sought after [16]. Providing a further picture of the requirements in the context of software engineering is the motivation for the research question $\mathcal{D}.\mathcal{RQ}_1$ of this dissertation.

There exist multiple other sources of data protection law within the EU. These are not in the core of this dissertation, but a terse overview is provided for context. Sources of law in general can be legislation, treaties, case law, doctrine, customary law, and fundamental principles [3]. For data protection law in the EU, the main sources are the founding Treaties, the CFFEU, the GDPR, the Police Data Protection Directive (2016/680), the ePrivacy Directive (2002/58) (and the ePrivacy regulation under negotiations) [3]. The data protection of EU institutions themselves is regulated by Regulation (EU) 2018/1725 [19]. The case law of Court of Justice of the EU and the decisions of the supervisor authorities provide interpretation [3]. Beyond these, there are many domain specific regulations that concern a certain part of data protection, such as the AI Act [20] or the proposal for the European Health Data Space [21] for instance. Furthermore, the EU-wide legislation is further complemented by nation-specific regulation.

In the academic sphere, data protection scholarship has many interdisciplinary approaches to it, as the topic often discusses personal data processing at the interface of regulation and software. The integration of privacy, data protection, and technology is discussed later on in Section 2.2.3 of this chapter. Even as a purely policy topic, the data protection has raised discussion [22; 23], and the topic is not "solved". It is clear, that providing *control* over their information and self-determination to data subjects is one of the main objectives of data protection legislation [24]. Within the context of software with a provider and and an outside user, there exists a (possibly fundamental) power and knowledge asymmetry between data subjects and providers. Lessening this asymmetry is one of the targets of data protection law [25].

Consent is one of the central pillars of data protection. In abstract, fully informed and free consent of a capable individual is a sound justification for nearly any action concerning them (including data processing). The tradition of informed consent is long standing in the scientific practice as well. However, the conditionals before the word carry a lot of weight. Many weaknesses in the practicality of consent have been raised [24]: often people do not have much choice whether to consent, users tend to click-through user interfaces blindly without *actually* consenting, and the revocation of consent is often ineffective. Whether consent as a concept is enough

for informational self-determination is an open question, but having a say is a part of it [26]. The present author would conjecture that an exhaustive solution to consent is not reachable, so long as we remain human, but there is plenty of room to improve the situation on many facets. This is something the dissertation contributes its small part into.

Personal data is defined in the GDPR succinctly as "any information relating to an identified or identifiable natural person ('data subject')" [13]. This is a particularly broad definition, as compared to for example 'personally identifiable information' of other jurisdictions. Even without taking it pedantically (any information of the world relates to all other information of the world), the definition has raised both discussion and confusion in practice [24; 27]. The definition being unclear might require case law, interpretation, and best practices to overcome. As an example issue, the same information could relate to multiple persons. For instance, the Internet protocol (IP) address is acknowledged as personal data, but subject access requests for them have been denied, in order to avoid data leaks from multiple persons having the same address [28]. This dissertation contributes in part to examining this space in $\mathcal{P}_5$.

The roles of *data subject* and *data controller* are also in the core of the GDPR. It should be noted, that the GDPR serves both of these parties. Enabling free flow of data is a major objective of EU data protection legislation, and this perspective is useful to keep in mind when evaluating legislative decisions [22]. This has been further reinforced with the introduction of the Data Governance Act in 2022, which facilitates data reuse by the public sector and voluntary data altruism [29]. Serving both the data subjects and controllers can be seen as a conflict within the definitions of the GDPR. The controller sets the rules, defines the means of processing, not the subject. The law sets rules for the controllers – the leading role is clear, with defined responsibilities including documentation. On the other hand, requiring consent shifts some responsibility back to the data subject, which suffers from the aforementioned asymmetry. Increased *transparency* is offered as a remedy [22], and building upon this notion by providing tools to provide better transparency is one of the themes of this dissertation.

The future holds refinement of the data protection rules, if not through other means, then by case law. An early look into the enforcement is provided in $\mathcal{P}_4$ of this dissertation.

2.1.2 Privacy policies & governance

Privacy policies have a large role in many parts of this dissertation, namely $\mathcal{P}_2$, $\mathcal{P}_3$, and $\mathcal{P}_5$. In this section a discussion on the topic of privacy policies and the governance of them is provided.

Informing data subjects and acquiring consent for data processing are the central tasks in establishing a data subject–controller -relationship. In the context of a

common consumer-grade information system service, this is typically achieved by presenting the data subject candidate (e.g. a customer) with an *end user license agreement* (EULA) and a privacy policy, alongside with perhaps (e.g.) a further selection of consents for specific data processing purposes and forms for initial data to create an account. A privacy policy is, then, the vehicle for providing the information required by data protection regulation to the data subject. For the purposes of this dissertation, the *form* of the privacy policy is secondary – it could consist of several pieces of information spread throughout an installation process, or a video, or (as evident in the material of $\mathcal{P}_5$) tens to hundreds of pages of inscrutable text. It should also be noted, that privacy policies as a concept are common worldwide, with differing legislation setting the rules for their content, and they have existed long before the GDPR. To reiterate, this dissertation examines the issue within the context of the EU and the GDPR.

At a minimum, a privacy policy should contain the following items: the purposes of processing personal data, description of the data in question, potential data recipients, if and to whom data is transferred to, erasure conditions, and information about the processing itself, such as security measures [30]. In addition, the data subjects must be informed of their rights and the data controller in question [13]. It is a common criticism, that privacy policies are so long and complicated, that users do not read them [31]. This is arguably criticizing the wrong issue, considering that the policy is often *the* way a user is informed about the method of processing their personal data [32]. The more detail, the more information for the user – if they are in the position to receive it. This is not to defend a poorly drafted or poorly presented policy, which could easily result from the risk-minimizing incentives of data controllers writing defensively. In fact, a lot of effort has been put into finding better ways of presenting privacy policies [31; 33; 34]. Based on these, it is clear that summarizing is required for users to read and understand. On the other hand, advanced users, such as *The Usable Privacy Policy Project* benefit from more information in assisting the common users [35].

To be precise, it is not necessary that a data subject agrees to a privacy policy (which may be the case in other jurisdictions). The data subject must be informed of the data processing, and if the basis for the processing is consent, it is required that the data subject agrees to consent [13]. The information required for informed consent is provided in the privacy policy, as per the definition of privacy policy in this dissertation, so 'agreeing to a privacy policy' is a short-hand. Should the basis of processing be something other than consent, the data subject has other avenues to object to the processing. This is in contrast to an EULA, which typically outlines the conditions a service may be used under, and agreeing to it is a requirement. Whether data subjects actually give their *informed* consent in the context of agreeing to web service privacy policies is an ongoing concern, as there is evidence that even simple and user-hostile policies are not comprehended by a large portion of users [36]. The

exact user-experience of asking for consent has been shown to have a large impact on the outcome [37] (whether for the benefit of the user or a dark pattern).

Privacy policies are, by default, textual documents that contain the required information. *Privacy languages* seek to provide a formal, semantic model for privacy information. When dealing with information systems, formal privacy policy models enable composing, analysing and processing the data (without involving any heuristic tools). As this is often desirable, there exists a rich literature in formal privacy languages, with different properties. One way to classify privacy languages is with a policy focus or security focus [38]; the former contains languages that model privacy policies for various use cases, and the latter access control and service level agreement models. As for the purposes of the present dissertation investigating formal privacy policies, literature review lead to choose *Layered Privacy Language (LPL)* [39] as the semantics for this work. It is specifically designed to model GDPR-compliant policies with a data model for each information, and is thus apt. The present author conjectures that translation between complete GDPR-compliant privacy policy languages is a mechanistic matter, and thus the exact choice of semantics is unimportant.

As privacy policies of consumer services are often publicly available, they are a common target for different kinds of analysis in research [40]. It has been noted, that privacy policies are neither explicit nor specific enough [41]. In order to obtain informed consent, the data subject must understand the purposes data is processed for, and policies have been found vague [41; 23]. It is possible to validate whether a policy contains all the provisions of the GDPR [42], and has been shown that the advent of the GDPR had measurable impacts on the privacy policies online [43]. Coupled with technology analysis, analysing privacy policies enables finding discrepancies between the policies and actual functionality of services. These discrepancies have been found to exist particularly in analytics [44] and cross-border data transfers [45].

Compliance, if nothing else, is a motivation for a data controller for following data protection principles (see $\mathcal{P}_4$ for enforcement actions). The governance of the policies of an organization is a cross-cutting concern, for example, within the purview of a data protection officer (or a chief privacy officer [46]). Managing the issue as a whole, requires integration of data protection into the overall risk management, information governance, and compliance function of an organization [47]. There are frameworks for modeling enterprise privacy governance, such as LINDDUN [39], which go beyond privacy policies themselves with a risk-management approach – as the regulation itself is risk-based in its compliance evaluation. Checking a box to say an organization is compliant with the GDPR is not simple, since there are qualitative aspects to the data protection requirements. In addition, the required interfaces to the data subjects are part of the user experience, and the public image of the organization. From the engineering perspective, governance efforts using only policies is not exhaustively complete, and technological measures are required [48].

This notion will be further elaborated on in Section 2.2.3.

2.2 Software engineering

The first section of this chapter dealt with the concepts and regulation of data protection and privacy. Let us now turn our focus to *software engineering* and how data protection is implemented in that context. The research questions of this dissertation, $\mathcal{D}.\mathcal{RQ}_1$ through $\mathcal{D}.\mathcal{RQ}_4$, in line with the goals of the dissertation, ask about an *application* of a data protection issue *within* software engineering.

Software engineering is both the research discipline and the practice of developing and maintaining software in a systematic way. A program, a unit of software, is an infinitely malleable definition of machine instructions; of how should the computer react to any given inputs. In isolation, developing a program is a rather abstract practice, more akin to solving a mathematical problem than building an object. This idea, then, meets real life, and complexity ensues: dependencies, errors, invalid requirements, budgets, *et cetera*. Two unique aspects of software, as compared to the products the traditional engineering disciplines, is that software can be modified and copied to distribute at will [49].

The goal of systematic, disciplined practice, referred to by the term *engineering*, in place of "software development", for instance, comes from striving for consistency. This applies in multiple dimensions including predictability, quality, maintainability, or business value. Systemic thinking, best practices, and standards enable collaboration. Even if Dijkstra put it [50], somewhat provocatively, that software engineering has accepted as its charter "how to program if you cannot." Yes, as far as the present author is concerned, software engineering accepts that we cannot. That we build systems greater than any one man, on top of highly complex real-life infrastructure, balancing competing constraints, for organizations with thousands of stakeholders, and millions of customers.

Software development life-cycle (SDLC) refers to the entire process of the life of a program. The traditional view of the SDLC is a sequence of linear phases ranging from defining requirements, to design, to implementation, to validation and finally maintenance. Over the years, there have various different paradigms and frameworks for systematizing software development, each with their own characteristics and trade-offs [51]. As discussed, an unique property of software is its malleability; software can and does evolve with the changing needs of its users over the product life. With this expectation, we can model the divergence of software functionality from the needs of the users over time, and identify metrics: *shortfall* for the gap from functionality to users' needs, *lateness* for the delay of solving a new need, *adaptability* for the rate of change, or *inappropriateness* for the area of shortfall over time [51]. Even at this time, decades later, the problem of how to optimally develop software is not generally solved.

Since the turn of the millenium, the notion of agile software development has gained popularity, as spread, for instance, by *the Agile Manifesto* [52]. Agile software development comes in multiple flavors, from *Scrum* [53] to *extreme programming* [54], which each define unique operating models for SDLC management. The singular core of agile methods is to focus on *change*, on the ability to change software to meet user needs and deliver value expediently. This dissertation views the issues of data protection in software engineering through the lens of agile software development. It is not concerned with any particular agile methodology, and the results are mostly independent of the paradigm at all. This positioning is mainly for setting the scope for motivation, discussion, and arguments based on the results.

The SWEBOK, *the Software Engineering Body of Knowledge*, represents the accumulated knowledge of the entire field [2]. In order to promote a consistent view of the domain worldwide, the *IEEE Computer Society* has collected and published a Guide to the SWEBOK since 1999, with updates published as the field has progressed [2]. The guide organizes the body of knowledge into fifteen knowledge areas ranging from requirements to software economics, with hierarchical topics under those, which cover software engineering in its entirety. For the purposes of this dissertation, the topical knowledge areas are software requirements, design, and management and professional practice. These are elaborated in the following sections.

2.2.1 Requirements engineering

Software *requirements* are critical to the development process. If you do not know what is required of a program, it is impossible to solve the design, or verify whether it is correct, or fit for purpose. A requirement can express any need or constraint placed on software (or the software development process) that contributes to solving the real-life purpose of the software [2]. In practice, requirements are not fully or correctly defined, and working software can still be designed. The lightest requirement specification could be a single sentence statement what purpose of the program should be. On the other hand, in theory, the ultimately complete requirements specification is the source code of the software itself – rendering the distinction meaningless. Requirements are an abstract map to the definition of the software, and they can concern all facets of it and the process of developing it. Requirements can be expressed in any kind of way, and concern widely different levels of detail or stakeholders. Software requirements have been traditionally classified into the categories of functional and non-functional, over whether they define *what* the software should do or *how* it should do it [2] – though this distinction in practice has been questioned in recent discussion [55].

Uncertainty and volatility in software requirements leads into uncertainty in the entire software development life-cycle [56]. As mentioned previously, the goal of software engineering is to reduce uncertainty in the process. The same applies to

software requirements. This sub-discipline of software engineering is known as *requirements engineering* [2]. Among the topics within RE are: *requirements fundamentals* research the nature of the requirements themselves, the *requirements process* seeks to integrate the requirements into the overall software engineering process, *elicitation* is about collecting requirements, *analysis* organizes and for example detects conflicting requirements, *specification* documents requirements, *validation* attempts to verify that the specification is correct, and practical considerations and tools help realize these in real life [57; 58; 59].

Requirements engineering has been a distinct domain of software engineering research, at least since 1980s [60], when the discussion on verifying and validating software expanded to verifying and validating software requirements. The economics are simple: catching mistaken requirements early on in the process is cheaper than later, especially in the sequential waterfall model of development. Failures in requirements engineering cascade to failures of entire projects [61]. This perspective of systematically analysing requirements on a meta-level for completeness, consistency, feasibility, and testability can be viewed as a paradigm shift from simply viewing requirements for their contents. The techniques for verification and validation can simply be manual reading, checklists, or more developed techniques such as defect, scenario, or perspective based reading, for instance [8; 62]. The *quality* of requirements can be studied, from various meta-aspects such as text-length to whether the content considers all relevant concepts [63].

In this dissertation, a major focus on requirements engineering is in *regulatory compliance*, under which data protection falls into. Law imposes requirements to a software business, particulars depending on the operating domain of the business. These requirements can be obligations for functionality or processes, negative or positive rights, quality requirements, *et cetera*. There are multiple sources of law, for example the EU and national legislation, and case law, each of which contributes to the totality of requirements for a software [3]. These requirements can even feasibly be in conflict with one another, and the only way to *know* your software is in compliance of law is to have it evaluated by a supervising authority, whether by an audit or a court [64]. The designers, that is, the requirements engineers, of a software business, must therefore have the ability to take into account all relevant legislation in the requirements process of the SDLC. The scope of this dissertation is limited to support software with the *intention* to be compliant, even if it could be rational in some circumstances, for example, to intentionally ignore a law and consider fines a 'cost of doing business'. There is a large gray area, though, as mentioned, since software is incredibly specific in its data processing instructions and law is interpreted from general rules.

Over the years of requirements engineering research, as digitalization is ever more weighty in all aspects of life, the focus of the field has also shifted to the impact on society and industries [65]. In an abstract sense, legislation and market

forces do requirements engineering, but it is also a two-way street, where requirements engineering advances affect society. It has been argued, that the decisions made and stakeholders in requirements engineering evolve, to be more data and user driven [66]. As the stakes get higher, along with the impact of digital services, any failures (e.g. in data protection and security) generate political pressure, beyond just market forces, to regulate software systems [6]. The framework for this dissertation is to simplify the model into two inter-dependent activities. These the present author calls, for a lack of established terms, the *regulatory technology transfer activity*, and the *software service-usage-enforcement feedback activity*. Firstly, the latter of the two, based on existing research [6], represents the cyclic mechanism of the software industry providing services to the public (the design of which is defined by their requirements engineering). The public is affected by the software and, in case of negative effects, pressures the government to intervene. The government enforces rules on the software industry, and launches initiatives to develop better regulation. This, in turn, again prompts the industry to further develop software. The activity is indeed concerned with the practical implications of software requirements.

The other mechanism, again based on existing research [7; 8], the regulatory technology transfer activity, is the complementary societal interplay between the government, academia, and the software industry. To improve in their endeavours, it has been argued that humans have historically taken two archetypal strategies [7]. Firstly, the *path of science* observes, measures, and experiments on the world, resulting in a better understanding of the world and candidate solutions. Secondly, the *path of engineering* constructs and evaluates, resulting in better tools and validation [7; 8]. Further discussion on the philosophy of science is provided in Chapter 3. The software service-usage-enforcement feedback activity results in the need to develop socio-legal-technical solutions to the mentioned issues. The government creates regulation, taking input from both academia and the industry, to tackle particular negative effects of software. To comply, requirements engineering as the science, and requirements engineering as the practice interact to improve our understanding of compliance and its implications, resulting in standards, best practices, arguments, and such.

A diagram of this model is presented in Figure 2. This framing is the context that requirements engineering in this dissertation discusses, and thus links it both to the paths of science and engineering. This dissertation contributes to the academic discussion on both of the societal activities in question. The GDPR is a prime example of this feedback loop in action: we can see the development of privacy and data protection requirements evolve over time from mostly non-functional quality requirements to encompass more functional requirements, such as the mechanisms to implement the rights of data subjects to data portability.

There are established methods for systematic compliance requirements elicitation, which are based on *argumentation* [67; 68]. The process is interpretative and

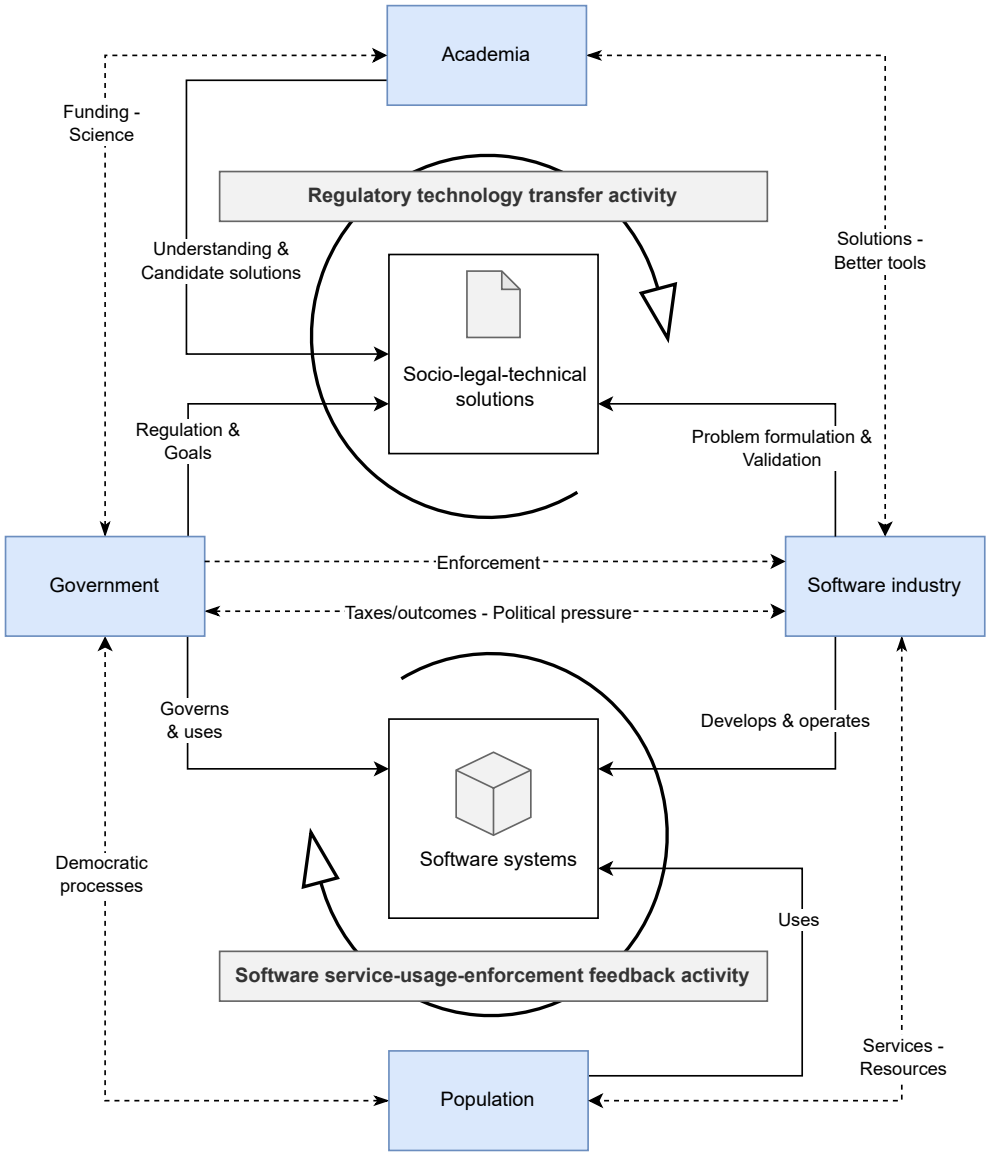

Figure 2. An overview of the societal context and stakeholders, their interactions and the two main activities considered in this dissertation. Synthesis of [6; 7; 8].

regulations may be complex, so there is generally no guarantee that a set of requirements is complete and sound in meeting the regulation in question. The role of engineering in this process is in systematization of the process. These approaches can, for instance, formally model arguments and thus reason about their relation to the requirements [67], or apply different argumentation schemes as templates to create re-usable patterns of compliance [68]. This kind of iterative and interpretative approach is deployed in $\mathcal{P}_1$ of this dissertation to elicit requirements from the GDPR. Beyond general compliance engineering in the abstract, plenty of requirements engineering research focusing specifically on GDPR compliance has been conducted since the introduction of the regulation [69]. These range from conceptual frameworks to modeling languages, with ongoing interest.

To conclude this section, a slight defence of the entire discipline of requirements engineering, or at least of the definition for the purposes of this dissertation, is in order. There is an ongoing conversation, as advanced for instance by Ralph [70], that the *paradigm* of the discipline (and other similar disciplines such as project management) is based on rationalism, and its results are contradicted by empirical research. That there are conflicting positions, such as *"developers use methods"* over *"developers do not empirically use methods as intended or at all"*, or *"functional requirements exist, in an objective reality, waiting for discovery"* over *"system requirements are an illusion motivated by our unwillingness to recognize epistemic uncertainty"* [70]. The essential difference between the paradigms is prescription over description. This dissertation entirely subscribes to the idea that software development is a complex activity and prescriptive models of it are bound to miss. There is room for nuance, however: when this dissertation discusses for example *requirements elicitation*, it is a description of particular activities a developer does during the complex, iterative process of development. It is not necessarily implied to be, for example, a distinct phase or something they explicitly *do*.

Another point to be made on the topic, is that software engineering is a social activity and much of the prescriptive methodology serves as a way to organize and coordinate people. Software requirements, too, are often a social construct. Systematic managing of them, again, is to facilitate coordination. Indeed, the next to incomplete requirements and moving targets, flaws in communicating requirements have been found the most common problems in requirements engineering [71]. On the other hand, beyond coordination, technical requirements themselves also have concrete business implications, considering a public procurement or regulatory compliance where the exact text can outweigh intent, for instance.

2.2.2 Software architecture

As the Guide to the SWEBOK puts it [2], *software architecture* as a term has evolved from simply concerning the set of structures needed to reason about relations and

properties of software elements, into a wider discipline of study. In its core, software architecture is about viewing a system from various perspectives and higher levels of abstraction. From this, common patterns and blueprints can be discovered, reasoned about, and re-used. Perry and Wolf influentially described architecture as the set of elements, forms, and rationale [72]. The last part, rationale, is integral, since software architecture is made for humans to reason about, and without the rationale, the choices made about the elements and forms in architecture design are not clear. An alternative simpler way to view architecture, is as the *significant* decisions made in the process of designing software [73]. A crucial skill is in knowing which decisions are the significant ones.

In general, the goals of software architecture are to prescribe constraints at an appropriate level, separate meaningful choices from aesthetics, express different aspects of software in a proper way, and analyse the dependencies and the consistency of software [72]. Again in line with the aforementioned definition of software engineering, having a systematic management of the chosen architectural constraints (and their rationale) results in the entire design process being more systematic. Viewing the structure of software from various abstraction levels and viewpoints aids in identifying the significant decisions. Without a proper architectural lens, it can be difficult if not impossible to reason about the properties of the software in question, or comprehensively identify and analyse the dependencies of different elements, which affects its implementation. In the end, a goal of a software architect is to identify present options in a situation, with the given trade-offs and the constraints.

Architectural *structures and viewpoints* describe and document different facets of a system [2]. A traditional way to model viewpoints of software is to divide them into five parallel architectural views (The 4+1 View Model) [74]. The logical view describes the functionality of the system for its users, modelled in class diagrams or entity-relationships, for example. The process view concerns the concurrency and other dynamic aspects of the system such as passing control, modelled in e.g sequence or activity diagrams. The development view considers the implementation in terms of subsystems or libraries, so that the work can be split across different developers. The physical view considers the mapping of the software onto hardware for execution. Finally, these four views are combined into use case scenarios, which are used to validate the integration of the different viewpoints. In modern developments, the description of different viewpoints have been developed into more formal standards. An influential example is the ArchiMate modelling language, which has a formal description of architecture views and viewpoints [75]: the purpose of the viewpoint mechanism is to address the concerns of particular stakeholders by providing a specific view (chosen representation of elements of interest) of the architecture in question [76]. Finally, in contrast to the formal modeling methods, it is appropriate to note that they are not necessary nor always practical, as opposed to lighter architecture descriptions [77].

As Booch describes it [73], a software system will have an architecture, whether it is intentionally considered or not. The so-called Conway's law is the observed effect that organizations tend to design system architectures that correspond to their organizational structure [78]. Not intentionally paying attention to the foundational design of a system, the result is an *accidental architecture*. This is not necessarily an issue, since building meaningfully novel software is an iterative process of exploration. The software engineering view into this distinction is turning accidental architectures into intentional ones, learning, and discovering patterns for the future. This viewpoint is in line with the critique of rationalization [70], that software architecture is not built in a single abstract phase of the design process. As a way of guiding the design of architecture in novel situations, where established rules are not present, *principles* are a way of setting reasonable boundaries that guide the process [79]. An influential example of these are the SOLID principles: a module should have a single responsibility, be open to extension and closed to modification, any module implementing an interface should be interchangeable, modules should not depend on methods they do not use, and that dependencies should be on abstractions, not on concrete implementations [79; 80].

Software architecture as a concept can relate to widely different scales of systems and levels of analysis. For instance, consider the difference in reasoning about the properties of passing messages between classes of a single program or of a distributed system of multiple independent modules. Both of these could even provide essentially the same functionality to an end user. As the scope of the topic at hand widens, even if it concerns software, the wider perspective is known as *information systems architecture* [81]. Even further, the systematic modeling of organization and communications of an entire business is known as *enterprise architecture* [82], with its own modeling methodologies such as The Open Group Architecture Framework (TOGAF) [83]. The present dissertation concerns quite a large range of different architecture levels in different parts of it, and many of the original publications can be viewed from both a software and information systems lens. Furthermore, there are no hard boundaries between these different levels of architecture.

In this dissertation, $\mathcal{P}_2$ and $\mathcal{P}_3$ concern *static analysis*. This is a software engineering technique to reason about the properties of a program without executing it. This is a way to gain thorough knowledge of a program, and find errors early on in the development process [84]. This provides a view into the architecture of a system and its properties, depending on the particular analysis. In this sense it is a *post hoc* method as opposed to deliberate pre-coding design, but having a view of the program structure is helpful for further development, and software is commonly developed iteratively. Therefore, even as static analysis requires something to analyse, it also serves as a method for rapid feedback iteration during development. In contrast to static analysis, $\mathcal{P}_5$ implements dynamic analysis via *black-box testing*, where only the user facing end result of a program is probed, without knowledge of

the internals. Both approaches have their uses and limitations. For instance, static analysis of the source code, naturally, requires access to the source code, which is not guaranteed for the user. Black-box testing, on the other hand, is more likely to miss functionality that is not apparent to the user [85]. Both fall under the class of *software verification and validation* [2], which can be considered its own sub-discipline of software engineering research and partly requirements engineering in terms of acceptance. Verification and validation can also be applied on the architecture level via static analysis; whether the implemented structure of program dependencies et cetera matches the planned one, for instance.

The prevailing software paradigm for the majority of this dissertation is software-as-a-service (SaaS), which are offered online as web services over application programming interfaces (APIs). SaaS in itself is closer to a business model rather than strictly an architecture, but the approach has technical implications nonetheless. While actual businesses differ in the details, the common pattern is that SaaS sells access to software, where the customer generally does not have access to the source code or the program – they use it through an API or another interface, or integration [86]. The term *web service*, in the context of this dissertation, refers to the abstract definition of software used over the internet over an API, as opposed to for example the Web Services standard, which specifies certain protocols for access [87]. The essential idea of a SaaS web service is (similar to the SOLID principles) that consumers of a service only depend on the interface of the service, the contract of it, rather than the implementation. With this abstract model, one can compose services and otherwise reason about their larger architectures with more clarity [88]. The term *semantic web* refers to a goal of moving the internet from generic interfaces to those with embedded semantic meaning [89]. Fully realized, it would enable automation on a more general scale, as opposed to the handcrafted workflows of the past [89]. While it remains to be seen if this aspiration would be made redundant with, for instance, artificial intelligence solutions overriding the need, there is value in simply specifying the API contract in a semantic way. This dissertation contributes to the semantic web conversation by coupling privacy policy purposes to web service schemas in $\mathcal{P}_3$.

The topics of this dissertation are inter-disciplinary and range from data protection, regulation, privacy, security, human-computer interaction, software business, to software analysis. Software architecture is the common unifying thread among these; each issue is viewed from an architectural lens and solutions offered in terms of software architecture.

2.2.3 Privacy engineering

To conclude the background exposition of the dissertation, having established the foundations, this section discusses the specific discipline of this dissertation: *privacy*

engineering. Privacy engineering is a relatively new field, establishing itself as a distinct research community in the 2010s, with the aim of developing theories, methods, techniques and tools for systematical improvement of privacy and data protection [5]. While the issues privacy engineering addresses are commonly intertwined with information systems and software engineering, the field has an explicit interdisciplinary perspective. For instance, in contrast to pure information systems research, privacy engineering does integrally study technology itself.

Privacy engineering advances the argument that we must consider privacy in every perspective related to software engineering, ranging from the technological infrastructure, to organizational controls, and user experience design [5]. In this view, purely technological approaches are insufficient, and software ought to be viewed as sociotechnical systems with an interdisciplinary lens. Research should focus on systematizing and generalizing solutions, and on integrating them into daily engineering practices, in order to make an impact. There are plenty of techniques, which need to be evaluated in different social, organizatorial, technical, or legal contexts to be fully realized [5]. Software often operates in various different and changing regulatory environments, which highlights the need for adaptability [90].

The threats to privacy are varied, and they are best defined from the perspective of data subjects and information output from them, as Solove categorizes on his taxonomy of privacy threats [91]. For an abstract and benign example, information is collected from a data subject, held and processed by a data holder, further disseminated to third parties, and finally fed back to the data subject in a generic way. Each of these categories of processes have their specific threats to privacy. For instance, in information collection, the data subject can be under surveillance or interrogated for private information. In information processing, data can be aggregated to a privacy breaking level by combining different data sources, identified by, for example, linking pseudonymous data to a person, stored insecurely, misused for secondary purposes, or a data subject be excluded from information about processing. Dissemination threats include breach of confidentiality, disclosures, exposures, increased accessibility, blackmail, distortion, and appropriation. Finally, invasions into people's private affairs include intrusions of privacy and decisional interference. Considering these threats from the data protection perspective, they can originate from outside actors or inside actors acting against instructions, which is in the realm of information security and privacy. While privacy engineering is concerned with implementing security in an integrated way, the other inside threat perspective is more interesting for the purposes of this dissertation – whether a designed policy and software solution presents risks of these threats to privacy even when used as intended.

An early influence for the developing privacy engineering field were Spiekermann and Cranor [92], who articulated the concepts of *privacy-by-architecture* and *privacy-by-policy* as a distinction between different privacy approaches. In practice these are a spectrum, but the ultimate goal of privacy-by-architecture is to engineer a

software system such that there is no need to collect personal data, or minimal data beyond, for example, pseudonymous tokens. On the other end of the spectrum, one provides the minimal privacy features that allow a user (for example) to consent to data processing by your policies; there is little privacy engineering done. Between the extremes, there is plenty of space for hybrid approaches, where personal data is (or must be) processed and its privacy and data protection can be improved with the methods, techniques, and tools privacy engineering studies. This is where the present dissertation is situated: pushing privacy-by-policy technologies and development further towards the privacy friendly direction. As the GDPR requires data controllers to take into account the *state of the art* when considering their technical measures in data processing, research in methodologies could have leverage in pushing the industry ahead [93].

As discussed previously, the foundational decisions about software are costlier to change than specific modifications. These decision include designs such as privacy-by-architecture patterns. Within privacy engineering, this notion was exemplified by Cavoukian in her principles of *privacy-by-design* (PbD), which means considering privacy from the start of any initiatives, with the goal of raising the bar for data protection worldwide [94]. These principles include proactivity over reactivity, privacy as the default, privacy embedded into design, full functionality with positive sum, securing data for its entire life-cycle, visibility and transparency, and respect for user privacy. These sparked discussion including criticism for being vague or not practically implementable [95; 96]. For instance, the simple principle of data minimization was suggested as a more practical principle [95]. Vagueness is a challenge in the sense that since no concrete requirements were set, anyone could claim their process follows the principles [96], and these "by design" regulatory principles have been argued to lack enough substance [97]. Nevertheless, for instance, the GDPR references a similar approach by its "data protection by design and by default" requirement [13]. Van Rest et al. propose a more concrete definition of PbD, which focuses on systematically applying privacy and data protection design patterns during the whole life cycle of software [96].

There is a well-known gap in the privacy engineering research and privacy engineering implementing the state-of-the-art as an industry practice [98]. There are plenty of tools, methods, and techniques published – and privacy in itself is generally recognized as a laudable value. It has been observed that this is caused both by ignorance, lack of tools, and lacking enforcement [99]. The gap in research and practice is further exacerbated by another related, but different, challenge known as the *privacy paradox*. This is an observed societal effect where there is an apparent inconsistency in the privacy preferences expressed by people and their actual behaviour online [100], though it is not (like any human behaviour) black and white [101]. As can be inferred from the PbD discussion [102], abstract privacy engineering goals are not trivial to put into practice. This, combined with the privacy paradox, might

lead to misaligned incentives in reaching industry practice from research. It has been observed that organizational cultures affect developers' attitudes to privacy [103] and knowledge dissemination from the academia might still be lacking [104]. The present dissertation is a part of this conversation, and probably subject to the aforementioned limitations as well. However, the technology transfer and concrete application approaches taken are ways this dissertation seeks to mitigate the gap.

Let us then review some ways privacy engineers contribute concretely to software, PbD or data protection by design. The data protection impact assessment (DPIA) is the mechanism a data controller uses to perform and document risk analysis for new personal data processing within the GDPR regime. In order to manage it, a controller needs to describe and map the processing operations and identify threats in order to mitigate risks – the prerequisite of which is essentially a software architecture viewpoint of data flows [105]. This can be facilitated by different kinds of architectural frameworks, such as LINDDUN for privacy threat analysis [39], a meta-model of processing [105], security controls in design models [106], or other system description models [107]. It has been noted that in practice these descriptions and threat modeling are performed disjointly by separate actors in an organization, and thus not particularly tied to software architecture either [107]. As these modeling methods tend to be an added layer to software, it can be conjectured that this disjointedness plays a part in the aforementioned gap in research and practice. The operationalization of the legal requirements in practice is a challenge, though it can be managed with systematic steps [108]. The approach in the present dissertation ($\mathcal{P}_2$ and $\mathcal{P}_3$) contributes to remediating the issue by generating the models from the software itself via static analysis. This must be acknowledged as a trade-off, though, too: the policy depends on software rather than software on the policy – to the extent of the coupling in concern.

Other privacy engineering tools are, for instance, concrete privacy enhancing technologies (PETs) or design patterns. There is a large collection of these technologies as catalogued, for instance, by Heurix *et al.* as they assembled a taxonomy of different kinds of PETs [109]. The taxonomy considers seven primary dimensions along which technologies can be viewed: trust scenarios, aspect of privacy between identity, content and behaviour, aim, security foundation, data dimension, trusted third party involvement, and reversibility. These include, for example, anonymization techniques [110; 111; 112], steganography [113], group signatures [114], or performing searches on encrypted data [115], privacy in location-based services [116], *et cetera*. Privacy design patterns can be classified under strategies [117], tactics [118], which organize concrete patterns [119; 120; 121]. The methods of privacy engineering also have parallels to other regulated domains such as finance, which present their own control patterns to ensure audit compliance [64].

Considering the aforementioned gap in research and practice, the study of software engineering methodology with regards to privacy warrants attention. Applying

privacy engineering techniques in industry practice ought to be an integrated part of the software development process. As discussed previously, the perspective of this dissertation is the agile, iterative software development context. It has been argued that there have been three specific turns in software development, which challenge privacy governance [122]: the change to favor agile methodologies, the change to favor SaaS, and the change to favor cloud computing. The exact challenges Gürses et al. identify are *modularity*, *temporality*, and *capture*. Modularity refers to the challenge of systems being increasingly composed from different independent parts, temporality to the changing and ever-evolving nature of software, and capture to the software market dynamics in which influential services shape the privacy expectations. The research in this dissertation is situated to mitigate each of these challenges. The posited notion of shifting the responsibility of privacy policy generation towards software engineering teams (temporality), composable privacy policy purposes from SaaS (modularity), and the analysis of privacy strategies in SaaS markets (capture) contribute directly to addressing each of the challenges. Research has identified that there are some crucial non-functional requirements for privacy mechanisms [123], including the usability in real-world information systems, coherent integration into existing practices, developer-friendliness, and reasonable performance [123]. The work in this dissertation addresses each of these requirements. When it comes to software engineering methodologies, though, it pays to keep in mind the long acknowledged notion of *no silver bullet* – the optimal methodology depends on the particulars of any situation [124]. Furthermore, the development of an organization is under the effect of *path dependency* – the optimal method depends on the previous methods and solutions employed [125].

2.3 Summary of the research gaps and motivation

To conclude the chapter, this section summarizes the identified research gaps and motivations for the dissertation outlined.

$\mathcal{M}_1$ There is a need to clarify the requirements of the GDPR, especially from a concrete and technical perspective. The need for clarity is present both in the requirements engineering and enforcement actions perspectives.

$\mathcal{M}_2$ The temporality of fast iterative development is a challenge in privacy engineering. While there exists plenty of research on static and data flow analysis in general, the intersection is underexplored, especially from the perspective of empowering software engineers.

$\mathcal{M}_3$ Privacy policies are often in the domain of other departments of a software developing organization than the software engineering functions. It is recognized that the modularity of SaaS architectures requires composability, and there are privacy languages that can model this in privacy policies. Software modeling in this regard is either too abstract and detached from the development or too low-level and detached from the policy context, and this dissertation addresses the gap in both theory and practice.

$\mathcal{M}_4$ Capture in digital platforms results in risks to data protection for consumers. The ecosystem of mobile applications and platforms tends to leave gaps in the roles between different stakeholders, and data protection risks are present, especially in the interfaces between actors. The way influential platforms . The particular use case of device sharing represents a known phenomenon with a lack of knowledge of industry practices for data protection management. Illuminating the effects on consumers in edge-case scenarios could provide paths forward for both the industry and regulators.

3 Research Description

This chapter describes the research undertaken for this dissertation. The chapter is provided in two parts: Section 3.1 reports the research strategy, the research questions, and research methodologies. Section 3.2 reports the summarized results of the original publications, and their relation to the context of the dissertation.

3.1 Research design

3.1.1 Strategy and foundation

Research design in software engineering is not a trivial task. The object of study is not merely technology but the entire engineering practice, which then requires concepts from social science as well [126]. This subsection outlines the practical technology transfer research strategy chosen for this dissertation, with the further subsection elaborating deeper into the tactical level of the individual contributions.

The interaction and problem-solving of humankind with the world at large has been described as having followed two fundamentally distinct but complementary paths: the paths of science and engineering [7]. The scientific method observes things as they are, measures and experiments, generating new knowledge. Engineering creates new things, constructs re-usable tools to solve problems, and evaluates them. Yet, there exists a necessary interplay between the paradigms: tools enable better science, science enables better tools [7]. Viewing these paths as an analogy for the distinction between software engineering as a science and as a practice requires some simplification, but, with the caveats, is the positioning of this dissertation. Revisiting the path of engineering, the rigorous design and evaluation of artifacts (e.g., tools) is known as *design science* [127]. Research with the goal of producing industry relevant applied results is known as technology transfer [8]. As discussed in Chapter 2, shifting the perspective slightly towards both a more concrete problem domain and a wider lens, technology transfer in data protection engineering must account for society as a whole as a stakeholder.

Before going further into the research strategy, it is appropriate to briefly affirm the *ontological* and *epistemological* context of the dissertation. The position of this dissertation follows the tradition of scientific realism [128]: there exists a reality independent of human minds, which can be observed with scientific methods. Theories can be true or false, and falsifiability is the cornerstone of theory. Software

systems are artifacts that exist in the world, but they can be viewed from multiple perspectives and an exact definition is nontrivial [129]. Artifacts themselves do not have a truth-value; the concept of truth is applicable only to propositions about their properties [127]. In the data protection context of the dissertation, compliance is one of the properties sought. The truth-value of compliance is not absolute either, but it can be true that somebody judged a subject to be compliant or not at a particular instance. Reaching the decision of compliance is based on argumentation and fact-finding. Whereas basic science provides descriptive knowledge, applied science also generates instrumental knowledge about an art of practice – there is a difference between a software engineer, the practice of designing software, software engineering as the art, and software engineering as the applied science [130]. In light of the Is-Ought -gap as described by Hume, when generating instrumental knowledge about how a practice is best done, there is a risk of "smuggling" moral judgements into applied science [130] – particularly in the data protection domain of this dissertation where human rights are in question. Let it be noted, that where "ought" is written in this dissertation, it is in the sense of improving the state-of-the-art of data protection engineering, and not a principled claim of moral philosophy. While, speculatively, most might agree that data protection and other recognized human rights are morally good, that is incidental to this dissertation. That is to say: whatever the philosophical data protection principle is or becomes, data protection engineering strives to apply it efficiently. In general, this dissertation follows a common sense philosophy of pragmatism [131] and rejects semantic games; terms are useful only as far as they have an instrumental value in making a distinction.

The strategy for this dissertation is the technology transfer model, albeit the implementation is iterative and tackles several different facets of data protection engineering. The strategy is an *integration* between the two gaps of academia and industry, and of science and engineering. The way this is approached, as the distinctions are fundamental and ought to be delineated, is by first dividing the work into individual work packages representing a single paradigm, and then arguing that the whole is greater than its parts.

The main foundation of this dissertation is based on the synthesis and expansion of two frameworks: the *empirically-based technology transfer* model and the *three cycle view of design science research* [8; 132]. Wohlin describes the empirically-based technology transfer model [8], in which industry identifies issues, academia formulates them as research problems and develops candidate solutions, which are finally validated and released as solutions in the industry. This dissertation extends the model with the socio-legal-technical dimension with the regulatory context in the data protection domain, as described in Chapter 2 – resulting in another feedback loop, where the research and regulation affect one another and the resulting industry solutions. To integrate this model into actual research strategy, the technology transfer model is overlaid with the three cycle view of design science research as proposed

by Havner [132]. The problem formulation stage is grounded in the relevance cycle in applied requirements engineering, the socio-legal-technical solution development is approached with the design science build–evaluate -cycle, and the development is grounded in theory in the rigor cycle. Finally, the last parts of the dissertation follow the traditional hypothetico-deductive scientific method in observing the effects of the GDPR on the industry. This research strategy is visualized in Figure 3. To emphasize, this is the foundation of the strategy; in practice, the process undertaken has many parallel instances at varying stages of maturity, which is further elaborated on in Section 3.1.3. Furthermore, the actions by the public sector are simplified in the model, but the aim is to highlight three interdependent effects of the ecosystem: regulation is an important source for data protection requirements engineering, data protection research affects policy development, and the final arbiter whether a solution meets the regulation is the court.

Some research has been published on the success factors of the technology transfer strategy [133; 134]. Garousi et al. surveyed root causes for low relevance and ways to improve them [133], which ought to be considered when reviewing the strategy. The first two causes for a lack of relevance were [133] *"(1) researchers having simplistic views (or wrong assumptions) about SE in practice"* and *"(2) lack of connection with industry."* The author of the present dissertation is fortunate enough to act in a dual role with the industry for the duration of the research, mitigating these issues. The third root cause was *"(3) wrong identification of research problems"*, which means working on problems which the industry does not find relevant, has been considered in the research design. Garousi et al. suggest using more practical research methods and problems, which this dissertation aims to be an example of.

Wohlin and Aurum describe a decision-making structure to selecting a research design for empirical software engineering [126]. These are revisited in Section 3.1.3 when discussing the methodology for the individual publications, but let us review the strategy phase decisions presented. Firstly, the decision structure makes an outcome choice between basic and applied research. This dissertation is mostly applied research, with a small basic research component. Secondly, there is a decision between inductive and deductive logic used. This dissertation employs both deductive and inductive logic in its different work packages, where the beginning of the pipeline visualized in Fig. 3 is theory-building research and the end is testing. Research purpose of the dissertation is in general exploratory and descriptive, with some aspects of evaluation. The dissertation employs both interpretivist and positivist research approaches in its different work packages, the former in the design science cycles and the latter in the evaluation cycles.

To summarize the strategy in plain terms, the dissertation seeks to improve data protection methods and understanding in four phases. Firstly, to formulate the problem via requirements engineering. Second, to design and validate solutions with industry collaboration. Third, to ground the solutions into theory and contribute to it.

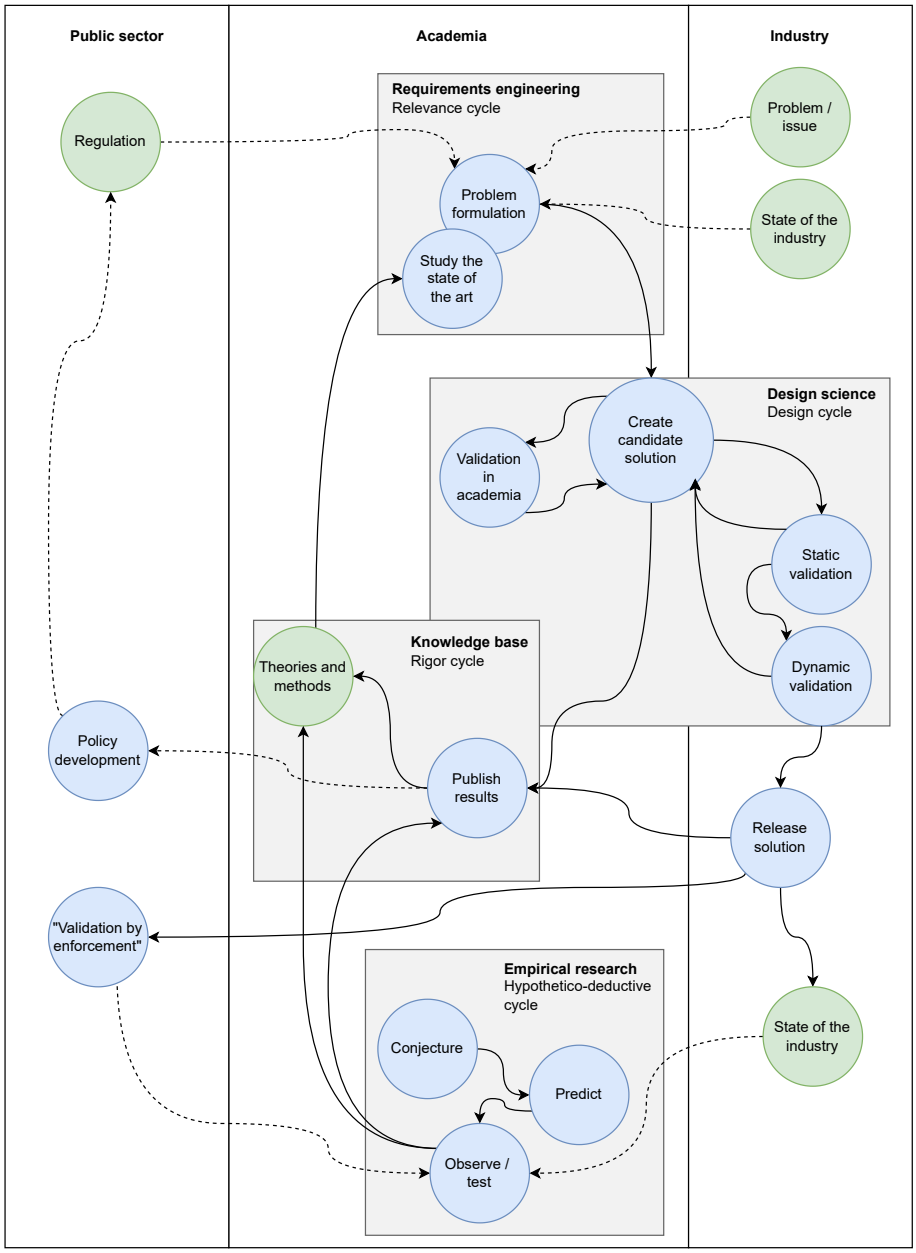

Figure 3. An overview of the regulatory technology transfer research strategy of this dissertation with societal context. Areas outlined in gray represent the scope of the dissertation. Synthesis of [8; 132].

And last, to observe the effects of the GDPR on the industry via empirical research.

3.1.2 Research questions

To contribute towards the two goals of the research, Improving the understanding of the GDPR and its implications ($\mathcal{G}_1$) and Improving data protection methods for software and system architectures and development processes ($\mathcal{G}_2$), from the viewpoint of software engineering practitioners, four research questions were defined on the operational level. These questions were met with five work packages, the original publications $\mathcal{P}_1$ through $\mathcal{P}_5$. What follows is the description and rationale for each question.

$\mathcal{D}.\mathcal{RQ}_1$ **What does the GDPR require of software-as-a-service technologies and their providers, and what are the implications of these requirements?**

At the date this dissertation research project began, the GDPR had recently came into force. The initial task of the dissertation is to set the context and map the main technical requirements from the GDPR, as motivated by $\mathcal{M}_1$. Beyond setting the context and initial requirements, the first publication $\mathcal{P}_1$ also sets the limits for scope of the dissertation into architectures and methods of business-to-consumer, software-as-a-service software engineering businesses. This question is one of the central ones of the dissertation, and thus multiple publications provide information into it. As the research was commenced over the period of several years, the later requirements and implications sought in the further publications are more nuanced and specific, as more knowledge is built over the research period. Publication $\mathcal{P}_4$ provides a view into the first enforcement actions over GDPR infringements, asking which GDPR articles are referenced in the enforcement cases and how the fines vary across the articles. Publication $\mathcal{P}_5$ provides a regulatory–ecosystem perspective into the question by examining the specific issue of device sharing and its manifestation in privacy policies and consumer device platforms.

$\mathcal{D}.\mathcal{RQ}_2$ **How can meeting GDPR requirements in software be improved from the perspectives of regulators, end-users, and in terms of software architecture patterns?**

This question is closely related to $\mathcal{D}.\mathcal{RQ}_1$ and also motivated by $\mathcal{M}_1$, but its focus is gaining insight into the current status of the industry. Several of the original publication research questions can be viewed from both perspectives; not only is it itself valuable to know the as-is status of industry solutions, in addition the results provide indications on what the GDPR actually requires in practice. For instance, publication $\mathcal{P}_5$ seeks answers from the policy documents of service providers, which indicate

how they handle certain data protection issues in the contract with their users. This could also reveal how ecosystem capture influences data protection solutions in the industry in line with $\mathcal{M}_4$. On the other hand, we can gain direct measurements of the challenges of the industry, at least from the perspective of the regulators, from the enforcement actions with administrative fines in $\mathcal{P}_4$.

$\mathcal{D}.\mathcal{RQ}_3$ **In what ways can information about personal data processing be represented within programming language type systems, and how can such representations be applied in software engineering practice?**

To meet $\mathcal{G}_2$ of the dissertation and provide technological advancements into software engineering practices, and the outlined research gaps in $\mathcal{M}_2$ and $\mathcal{M}_3$, this question statement is formulated. The motivation for the question follows from the influential and foundational thought of *privacy engineering*, that privacy ought to be built into the architecture of a system, as opposed to "privacy-by-policy" only [48]. It is a trivial observation that the state-of-the-art of practice had not, and has not, widely reached the privacy-by-architecture vision. This dissertation takes both a step deeper and a step simpler, and seeks to improve the issue not from policy but from the "engine room" – the type systems of the source code of the services. As opposed to entirely forgoing personal data processing via architectural decisions, the aim is to enable architectural decisions that could even remove certain classes of misprocessing entirely. Contributions to this question are provided in publications $\mathcal{P}_2$ and $\mathcal{P}_3$.

$\mathcal{D}.\mathcal{RQ}_4$ **How do data protection requirements align between business objectives, privacy policies, and systems and software architecture?**

This question is posed as the conclusion of the dissertation, where the perspective is shifted from software internals to higher level architecture. This question also ties all of the previous questions together with the motivations $\mathcal{M}_2$ and $\mathcal{M}_4$ of reaching a big picture of this exact intersections of domains. The first two questions seek an understanding of the domain, the third question provides technological contribution, and here $\mathcal{D}.\mathcal{RQ}_4$ takes a holistic angle and contributes a mental model, thus wrapping up the dissertation.

The original publications $\mathcal{P}_1$ through $\mathcal{P}_5$ contain the research which bridge the above questions into concrete contributions. What follows is the description of these publication's research questions ($\mathcal{P}.\mathcal{RQ}$s), which reside on the metric level as per the GQM-framework, and their relation to the dissertation research questions they provide information for. This mapping can also be viewed in summarized form in Table 1.

$\mathcal{P}_1$ **The General Data Protection Regulation: Requirements, Architectures, and Constraints**

$\mathcal{P}_1.\mathcal{RQ}_1$ *What practical constraints SMEs operating with SOAs typically face when implementing solutions for complying with the GDPR's new regulatory demands?*

$\mathcal{P}_1.\mathcal{RQ}_2$ *What requirements does the GDPR imply for SOAs operated by SMEs, and how these can be elicited?*

$\mathcal{P}_1.\mathcal{RQ}_3$ *How the requirements elicited from the GDPR can be reasonably implemented in SOAs under the practical constraints that many software development SMEs face?*

The metrics of $\mathcal{P}_1.\mathcal{RQ}_1$ through $\mathcal{P}_1.\mathcal{RQ}_3$ published in $\mathcal{P}_1$ contribute to the dissertation research questions $\mathcal{D}.\mathcal{RQ}_1$ for each of them and additionally to $\mathcal{D}.\mathcal{RQ}_2$ for $\mathcal{P}_1.\mathcal{RQ}_3$. The rationale is that the constraints outlined in $\mathcal{P}_1.\mathcal{RQ}_1$ provide the context from the perspective of which the requirements are viewed from. $\mathcal{P}_1.\mathcal{RQ}_2$, directly in the domain of $\mathcal{D}.\mathcal{RQ}_1$ and contributing to $\mathcal{D}.\mathcal{RQ}_4$, elicits the actual requirements of the GDPR in the chosen framing. Then, making use of the previous results, $\mathcal{P}_1.\mathcal{RQ}_3$ provides further understanding to the implications of the requirements relating to the implementation in software architectures.

$\mathcal{P}_2$ **Annotation-Based Static Analysis for Personal Data Protection**

$\mathcal{P}_2.\mathcal{RQ}_1$ *What parallels there are between those static analysis solutions designed for security and those that seek to improve privacy and data protection?*

$\mathcal{P}_2.\mathcal{RQ}_2$ *How the practical state-of-the-art of static analysis solutions can be applied in the context of the GDPR's data protection requirements?*

The second work package $\mathcal{P}_2$ provides two metrics to the questions $\mathcal{D}.\mathcal{RQ}_2$, $\mathcal{D}.\mathcal{RQ}_3$, and $\mathcal{D}.\mathcal{RQ}_4$. Firstly, $\mathcal{P}_2.\mathcal{RQ}_1$ applies the understanding of static analysis tools in the security domain into data protection and thus reviews the current status, and provides information to improving the practice. Second, $\mathcal{P}_2.\mathcal{RQ}_2$ directly contributes a new method for as a part of $\mathcal{D}.\mathcal{RQ}_3$ and for $\mathcal{D}.\mathcal{RQ}_4$.

$\mathcal{P}_3$ **Extracting LPL privacy policy purposes from annotated web service source code**

$\mathcal{P}_3.\mathcal{RQ}_1$ *How can we model personal data processing for web services using LPL?*

$\mathcal{P}_3.\mathcal{RQ}_2$ *How to automatically extract the model's data from web service source code?*

This publication is mainly providing its metrics to $\mathcal{D}.\mathcal{RQ}_3$ as both of its publication research questions $\mathcal{P}_3.\mathcal{RQ}_1$ and $\mathcal{P}_3.\mathcal{RQ}_2$ investigate modeling personal data processing in software. The question setting take the form of a follow-up to $\mathcal{P}_2$, extending the perspective from verification and validation to generating data for privacy policies. The first metric $\mathcal{P}_3.\mathcal{RQ}_1$ additionally contributes information into $\mathcal{D}.\mathcal{RQ}_4$ with its general architectural discussion.

$\mathcal{P}_4$	**The GDPR enforcement fines at glance**
$\mathcal{P}_4.\mathcal{RQ}_1$	*Which GDPR articles have been actively referenced in the recent enforcement cases?*
$\mathcal{P}_4.\mathcal{RQ}_2$	*Do the enforcement fines vary across the articles referenced in the enforcement decisions?*
$\mathcal{P}_4.\mathcal{RQ}_3$	*How well the recent GDPR fines can be predicted in terms of basic available (i) meta-data and (ii) textual traits derived from the enforcement decisions?*

The rationale for the metrics in $\mathcal{P}_4$ is that the GDPR enforcement actions provide information both about the actual requirements ("validation by enforcement" as named previously) for $\mathcal{D}.\mathcal{RQ}_1$ and of the industry status for $\mathcal{D}.\mathcal{RQ}_2$. The individual metrics of $\mathcal{P}_4.\mathcal{RQ}_1$ through $\mathcal{P}_4.\mathcal{RQ}_3$ all deal with the same issue from slightly varying perspectives.

$\mathcal{P}_5$	**Device Sharing Features: A study on software policy approaches and platform capabilities**
$\mathcal{P}_5.\mathcal{RQ}_1$	*What are the common assumptions and strategies employed by SPS providers to account for device sharing in their policy documents?*
$\mathcal{P}_5.\mathcal{RQ}_2$	*To what extent the model of RQ1 maps into current technical capabilities and limitations of the software platforms for device sharing use cases, and to what extent have application developers themselves implemented features to facilitate device sharing use cases?*

The final publication of the dissertation, $\mathcal{P}_5$, contributes metrics to multiple dissertation research questions and is motivated by $\mathcal{M}_4$. Firstly, $\mathcal{P}_5.\mathcal{RQ}_1$ provides further information into $\mathcal{D}.\mathcal{RQ}_1$ by collecting the issues present in policies, and more explicitly for $\mathcal{D}.\mathcal{RQ}_2$ by reviewing industry solutions to the issues. Both $\mathcal{P}_5.\mathcal{RQ}_1$ and $\mathcal{P}_5.\mathcal{RQ}_2$ contribute new information into $\mathcal{D}.\mathcal{RQ}_2$ and $\mathcal{D}.\mathcal{RQ}_4$. The rationale is that the specific topic of device sharing also reveals indications of more general causes and effects of the ecosystem and the industry, and how capture affects data protection solutions.

Table 1. The mapping of the original publication metrics to the dissertation research questions.

Goal	Motivation		Question	Metric
$\mathcal{G}_1$	$\mathcal{M}_1$	$\mathcal{D}.\mathcal{RQ}_1$	What does the GDPR require of software-as-a-service technologies and their providers, and what are the implications of these requirements?	$\mathcal{P}_1.\mathcal{RQ}_1$ $\mathcal{P}_1.\mathcal{RQ}_2$ $\mathcal{P}_1.\mathcal{RQ}_3$ $\mathcal{P}_4.\mathcal{RQ}_1$ $\mathcal{P}_4.\mathcal{RQ}_2$ $\mathcal{P}_4.\mathcal{RQ}_3$ $\mathcal{P}_5.\mathcal{RQ}_1$
$\mathcal{G}_1$,$\mathcal{G}_2$	$\mathcal{M}_1$,$\mathcal{M}_4$	$\mathcal{D}.\mathcal{RQ}_2$	How can meeting GDPR requirements in software be improved from the perspectives of regulators, end-users, and in terms of software architecture patterns?	$\mathcal{P}_1.\mathcal{RQ}_3$ $\mathcal{P}_2.\mathcal{RQ}_1$ $\mathcal{P}_2.\mathcal{RQ}_2$ $\mathcal{P}_3.\mathcal{RQ}_1$ $\mathcal{P}_3.\mathcal{RQ}_2$ $\mathcal{P}_4.\mathcal{RQ}_1$ $\mathcal{P}_4.\mathcal{RQ}_2$ $\mathcal{P}_5.\mathcal{RQ}_1$ $\mathcal{P}_5.\mathcal{RQ}_2$
$\mathcal{G}_2$	$\mathcal{M}_2$,$\mathcal{M}_3$	$\mathcal{D}.\mathcal{RQ}_3$	In what ways can information about personal data processing be represented within programming language type systems, and how can such representations be applied in software engineering practice?	$\mathcal{P}_2.\mathcal{RQ}_1$ $\mathcal{P}_2.\mathcal{RQ}_2$ $\mathcal{P}_3.\mathcal{RQ}_1$ $\mathcal{P}_3.\mathcal{RQ}_2$
$\mathcal{G}_1$,$\mathcal{G}_2$	$\mathcal{M}_2$,$\mathcal{M}_4$	$\mathcal{D}.\mathcal{RQ}_4$	How do data protection requirements align between business objectives, privacy policies, and systems and software architecture?	$\mathcal{P}_1.\mathcal{RQ}_2$ $\mathcal{P}_2.\mathcal{RQ}_2$ $\mathcal{P}_3.\mathcal{RQ}_1$ $\mathcal{P}_5.\mathcal{RQ}_1$ $\mathcal{P}_5.\mathcal{RQ}_2$

3.1.3 Data and methodology

This section outlines the research methodology applied and data used for each work package $\mathcal{P}_1$ through $\mathcal{P}_5$ included in this dissertation. As per the strategy outlined previously, the main approach is requirements engineering in combination with design science and basic research. The following headings for each publication elaborate these on the concrete level, and summary tables Table 2 and Table 3 are provided at the end of the section.

$\mathcal{P}_1$ The General Data Protection Regulation: Requirements, Architectures, and Constraints

The first publication included in the dissertation approaches its three research questions each with different research methods. The research was performed on three partly overlapping phases. Firstly, the context was delineated with observations of a case study small-medium enterprise (SME) SaaS-provider subject. Secondly, the technical requirements of the GDPR were requirements engineered for the chosen context. This was performed in the argumentative requirements engineering fashion.

Finally, the two previous phases were synthesized in a design science study creating example software architectures for the requirements and the trade-offs analysed. The data used is composed of two contrasting sets: the main part is the archival data of the GDPR regulatory text itself, and the contextualising case study data is reflection-on-action [135] observational notes. The data analysis was thematic for $\mathcal{P}_1.\mathcal{RQ}_1$ and grounded theory for $\mathcal{P}_1.\mathcal{RQ}_2$. The rationale for the chosen methods is to gain a strongly practical result on the requirements of the GDPR, in line with the success factors for technology transfer [133], yet provide the analysis based on the actual text of the regulation, instead of a secondary perspective. This research can be classified as applied qualitative research.

$\mathcal{P}_2$ Annotation-Based Static Analysis for Personal Data Protection

The second publication included in the dissertation has a two-pronged approach. The first prong for $\mathcal{P}_2.\mathcal{RQ}_1$ is a literature review on static analysis tools, with thematic analysis and synthesis, also building on $\mathcal{P}_1$. This can be classified as basic qualitative secondary research. The second prong for $\mathcal{P}_2.\mathcal{RQ}_2$ is design science research with two outputs: a method and a tool. These designs were verified in an industry environment and software metrics of the pilot evaluated. Literature review followed by design science research research is an appropriate method for developing and demonstrating the feasibility of a software concept. This can be classified as applied qualitative research.

$\mathcal{P}_3$ Extracting LPL privacy policy purposes from annotated web service source code

The third publication included in the dissertation expands $\mathcal{P}_2$ to a new direction: privacy policy data generation employing static analysis. This is reached with two different contributions of applied design science. Firstly, the theoretical framework of Layered Privacy Language is expanded to meet the SaaS context of $\mathcal{P}_1$. Then, this framework is implemented in a static analysis method for generating the data for privacy policies that is designed and validated in a case study. These designs were verified in an industry environment and software metrics of the pilot were evaluated. Desing science research is an apt method for these questions, as the aim is to demonstrate the feasibility of a concept by a validated example. Both $\mathcal{P}_3.\mathcal{RQ}_1$ and $\mathcal{P}_3.\mathcal{RQ}_2$ can be classified as applied qualitative research.

$\mathcal{P}_4$ The GDPR enforcement fines at glance

The fourth publication included in the dissertation is focused on basic research of the effects of the GDPR in contrast to the applied previous contributions. The data set is a public (non-governmental) repository of GDPR enforcement fines data, that

is, archival data, which is analysed after due pre-processing with statistical methods from three different angles. Articles referenced in enforcement cases and amounts of fines are explored with descriptive statistics, and ordinary least squares are used to test the correlation of the fines between the cases. Finally, three regression methods were compared in their prediction of fines performance for six different models. The rationale for selecting these quantitative methods as opposed to thematic analysis, for instance, is to attach the study stronger to the actual enforcement result (that is, the fine). This research can be classified as basic quantitative research.

$\mathcal{P}_5$ Device Sharing Features: A study on software policy approaches and platform capabilities

The fifth and final publication included in the dissertation researches the effects of the GDPR on the industry by focusing on a specific use case, device sharing, in a mixed-methods study. The study was performed in two distinct sequential phases. The first phase, for $\mathcal{P}_5.\mathcal{RQ}_1$, analyses the policy documents of a representative sample set of mobile applications and web services, that are the end user license agreements and privacy policies. These can be classified as archival data [126]. The content is thematically analysed with a systematic mapping process, aiming to create a model of commonly used assumptions and strategies across the different services. This first phase can be classified as basic qualitative research. The rationale for choosing this method is to find the general strategies across the sample set, and the texts necessarily require human interpretation. The second phase, for $\mathcal{P}_5.\mathcal{RQ}_2$, applies the results of the first in employing requirements engineering and design science in order to model the device sharing strategies conceptually, as requirements and concretely testable use cases. Further picture of the device sharing strategy implementations and validation for the model is then provided with an experiment that black-box tests the models against the sample set of software. This method was selected for its ability to generate and test new concepts for the issue. The second phase can be classified as applied qualitative research.

Table 2. Research outcome, process, data collection method and data analysis method as per Wohlin and Aurum [126], for $\mathcal{P}_1$ through $\mathcal{P}_5$.

Metric	Outcome	Process	Methodology	Data collection	Data analysis
$\mathcal{P}_1.\mathcal{RQ}_1$	Applied	Qual.	Case study	Observation	Thematic analysis
$\mathcal{P}_1.\mathcal{RQ}_2$	Applied	Qual.	Case study	Archival	Grounded theory
$\mathcal{P}_1.\mathcal{RQ}_3$	Applied	Qual.	Design science	$\mathcal{P}_1.\mathcal{RQ}_1$ and $\mathcal{P}_1.\mathcal{RQ}_2$	Synthesis
$\mathcal{P}_2.\mathcal{RQ}_1$	Basic	Qual.	Secondary research	Literary review	Thematic analysis
$\mathcal{P}_2.\mathcal{RQ}_2$	Applied	Qual.	Design science	Software metrics	Descriptive
$\mathcal{P}_3.\mathcal{RQ}_1$	Applied	Qual.	Design science	Software metrics	Descriptive
$\mathcal{P}_3.\mathcal{RQ}_2$	Applied	Qual.	Design science	Software metrics	Descriptive
$\mathcal{P}_4.\mathcal{RQ}_1$	Basic	Quant.	Case study	Archival	Statistical
$\mathcal{P}_4.\mathcal{RQ}_2$	Basic	Quant.	Case study	Archival	Statistical
$\mathcal{P}_4.\mathcal{RQ}_3$	Basic	Quant.	Case study	Archival	Statistical
$\mathcal{P}_5.\mathcal{RQ}_1$	Basic	Qual.	Case study	Archival	Thematic analysis
$\mathcal{P}_5.\mathcal{RQ}_2$	Applied	Qual.	Design science & Case study	Experiment	Descriptive

Table 3. A summary of datasets used in the research in $\mathcal{P}_1$ through $\mathcal{P}_5$.

Domain	Dataset type	Collection	Publications
Regulators	The GDPR	Archival	$\mathcal{P}_1$
	GDPR enforcement actions	Archival	$\mathcal{P}_4$
Industry	Web service code bases	Case study	$\mathcal{P}_2$, $\mathcal{P}_3$
	Case company practices	Case study	$\mathcal{P}_1$
	Privacy policies and EULAs	Archival	$\mathcal{P}_3$, $\mathcal{P}_5$
Academia	Primary studies	Literature review	$\mathcal{P}_2$

3.2 Results

This section summarizes the results of the original research contributions $\mathcal{P}_1$ through $\mathcal{P}_5$. The results are organized within the dissertation framework by their $\mathcal{D}.\mathcal{RQ}$s.

3.2.1 $\mathcal{D}.\mathcal{RQ}_1$: The requirements of the GDPR

This dissertation research question, $\mathcal{D}.\mathcal{RQ}_1$ is formulated as *"what does the GDPR require of software-as-a-service technologies and their providers, and what are the implications of these requirements?"* The question was contributed to by three publications: $\mathcal{P}_1$, $\mathcal{P}_4$, and $\mathcal{P}_5$. The main contribution is the requirements elicitation of the technical requirements of the GDPR performed in $\mathcal{P}_1.\mathcal{RQ}_2$. This is supported by metrics from GDPR enforcement data, which elucidate the relative importance of each requirement. The results of the case study in $\mathcal{P}_1.\mathcal{RQ}_3$ and the policy analysis of $\mathcal{P}_5.\mathcal{RQ}_1$ provide information about the practical implications for the second part of the question. Further information on the practical implications of the requirements.

The result for $\mathcal{P}_1.\mathcal{RQ}_2$ is a systematically requirements engineered map of GDPR articles to technical requirements. Nine high-level requirements were identified, with

related sub-requirements that specify further aspects of them. The requirements were analysed in the context of the case company and the constraints of its operating environment outlined in $\mathcal{P}_1.\mathcal{RQ}_1$. Enumerated, the requirements are: R1: *system security and privacy*, R2: *data minimization*, R3: *consent control*, R4: *data traceability*, R5: *user access*, R6: *data rectification*, R7: *data erasure*, R8: *data restrictions*, and R9: *physical location of data*. These contain both functional and non-functional requirements. For instance, the requirement R1 contains traditional information security needs that originate from articles 25 and 32. On the other hand, the consent requirements for R5 imply entirely new functionalities that extend to the end user interface. This result is visualized in Fig. 4. The deduction can be made from the result, that it is possible to elicit a set of technical requirements from the text of the GDPR (though the correctness was not validated in court), but that compliance does require special consideration and changes to existing systems, and that the elicitation used case specific context.

The next metrics for $\mathcal{D}.\mathcal{RQ}_1$ are presented in $\mathcal{P}_4$. The three $\mathcal{P}.\mathcal{RQ}$s represent different statistical views on GDPR enforcement, which provide indirect information about the frequency and priorities of the enforcement of the regulation. Unsurprisingly, the articles for the data protection principles and lawful purposes were most commonly referenced in the enforcement cases, as illustrated in Fig. 5, but many other articles have been referenced as well. The term frequency statistics of the textual content identify security issue related terms as the common themes, which possibly referencing data breaches as the cause for an enforcement action. There were no strong correlations between the articles referenced and the amounts of fines, however – the country of origin and the year of the decision had more significance. This was contrary to initial expectations. The prediction model trained for $\mathcal{P}_4.\mathcal{RQ}_3$ did have predictive power, and the model with access to textual case content outperformed models with only meta-data. These results from $\mathcal{P}_4$ corroborate the notion that the GDPR is viewed as a holistic regulation over specific check-box-like requirements. The principles must be followed, and the accountability criterion requires the ability to demonstrate compliance.

The final metric for this question is $\mathcal{P}_5.\mathcal{RQ}_1$, which explored the privacy policies and EULAs of a sample set of software products and services. The result was a systematic map of strategies software providers have approached different issues regarding device sharing with. Overall, 18 different strategies were coded in the sample, and these were further categorized into five themes: *Access to software product or service*, *Age gating*, *Credentials*, *Multiple users*, and *Denial*. It was observed that there were strategies in multiple themes that were interpreted to be based on the GDPR requirements. For instance, the age gating theme as illustrated in Fig. 6 or parental consent requirements display influence of the GDPR requirements in the policies. This result showed variance in the strategies data controllers take (for a niche personal data processing issue), which could indicate ambiguous regulation

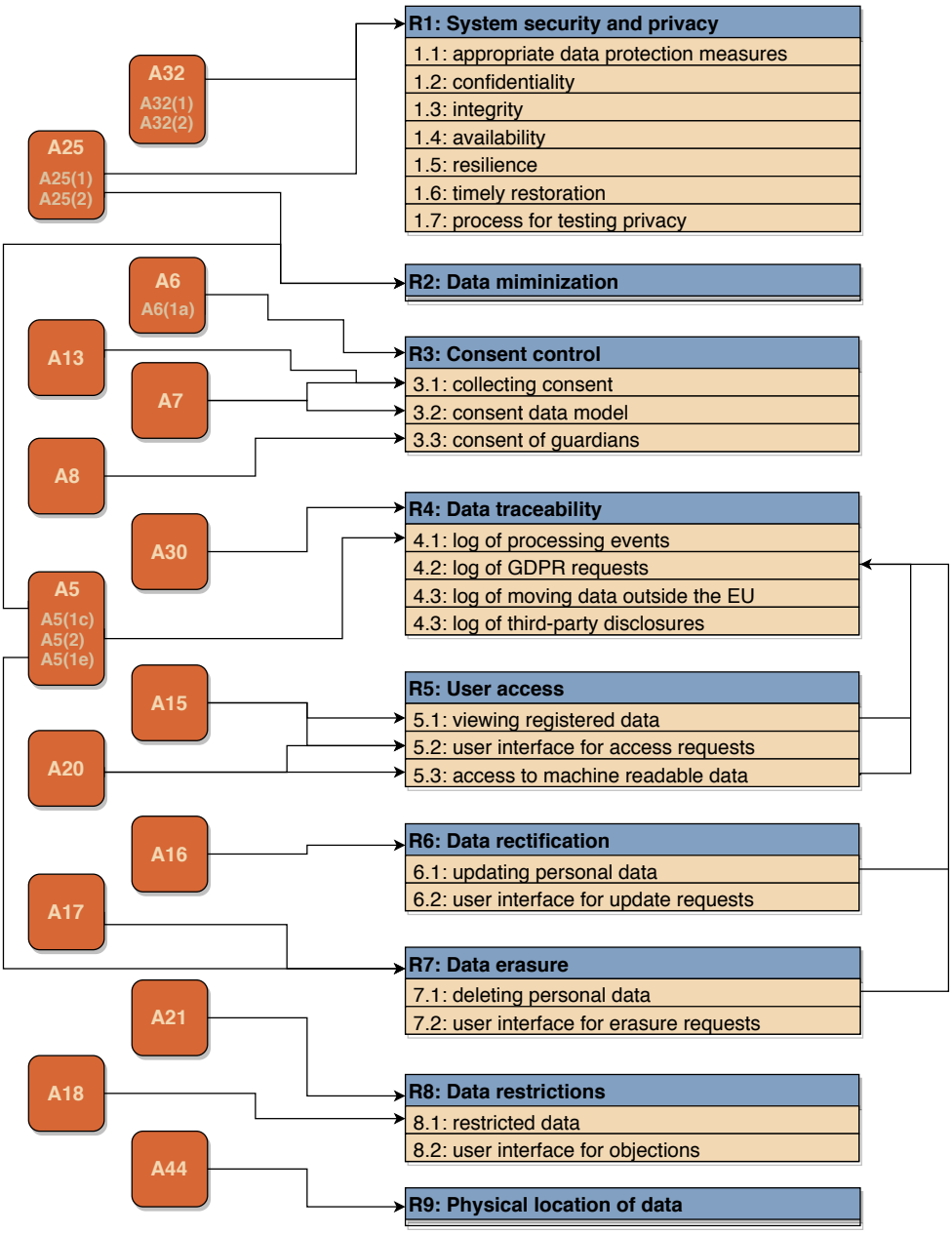

Figure 4. Elicitated requirements from the GDPR (from $\mathcal{P}_1$).

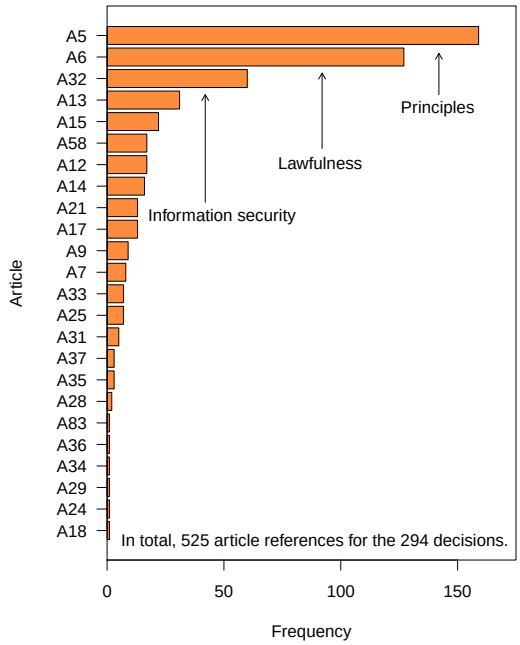

Figure 5. Articles referenced in GDPR enforcement cases (from $\mathcal{P}_4$).

among other causes.

As to the direct answer to the dissertation research question itself, *"What does the GDPR require of and what are the implications for software-as-a-service technologies and their providers?"*, it requires a formulation that synthesizes the previously enumerated results into a dissertation. We have data on the requirements engineering process for a case study, enforcement data indicating issues across the board, particularly in the high-level principles, and policy analysis results finding imaginative industry solutions. These all demonstrate the multi-faceted nature of GDPR compliance. With that justification, this dissertation makes the following claim ($\mathcal{C}_1$):

$\mathcal{C}_1$ The GDPR requires organizations to raise the baseline standard of data protection in industry software-as-a-service systems, through an undertaking that is inherently interdisciplinary. Although the regulation can be translated into technical requirements with relative clarity, compliance extends beyond information security. Regulators place emphasis on adherence to the GDPR's fundamental principles, including lawfulness of processing and accountability. This ought to be considered in software engineering processes.

While the enumerated general claim is a non-problem-solving result in the sense that the result merely emphasizes complexity, the individual $\mathcal{P}.\mathcal{RQ}$ results from the

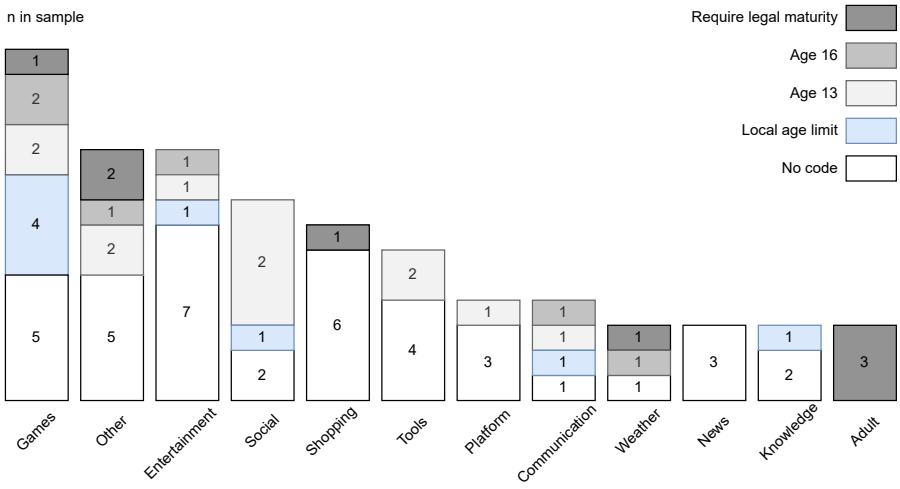

Figure 6. Example of strategies identified in privacy policies and EULAs within the theme *Age gating* (from $\mathcal{P}_5$).

original publications that contributed to the $\mathcal{D}.\mathcal{RQ}_1$ themselves also provide practical contributions. The actual mapped requirements can be extrapolated to other contexts, the enforcement data and the policy analysis device sharing strategies can inform the industry.

3.2.2 $\mathcal{D}.\mathcal{RQ}_2$: Avenues for improvement

The second dissertation research question, $\mathcal{D}.\mathcal{RQ}_2$ was formulated as *"how can meeting GDPR requirements in software be improved from the perspectives of regulators, end-users, and in terms of software architecture patterns?"* The question is closely related to the first one. The answer is a synthesis of the case studies and empirical industry studies, with a couple of auxiliary metrics. These contributions are present in the publications $\mathcal{P}_1$, $\mathcal{P}_4$ and $\mathcal{P}_5$.

Firstly, from the perspective of software architecture patterns, $\mathcal{P}_1.\mathcal{RQ}_3$ demonstrates practical and concrete changes with a design science method to a case study service oriented software architecture (SOA), for a transaction processing system. The pattern chosen was a collection of micro-service components, which implement data subject request requirements, personal data event logging, central consent and restriction control, and isolate core personal data from business logic services. Further technical measures were designed, including pseudonymization for staging environments and static analysis. These results designed for a case architecture do not, naturally, generalize universally, but there are some patterns to be outlined, which warrant consideration in similar circumstances. *Data subject requests as components*

includes setting up dedicated services to administrate data subject request based on meta-data, and their processing with a publish-subscribe -mechanism to propagate the implementation across the SOA. To reduce the risks in personal data propagation across the SOAs databases, especially in an environment with multiple data processing parties, the pattern of *isolating core personal data from business logic* was employed. This results in "zones of de-identification" [136] in areas of the SOA where particular personally identifying information is not needed, and follows the minimization principle. This draws parallels from software compartmentalization practices in information security, which aim to limit the impact of vulnerabilities [137]. The design also proposes a *centralized consent management service employing an authorization protocol* for checking specific consent, enabling fine-grained permissions to be defined.

Continuing with the solutions, $\mathcal{P}_5$ provides a synthesis of concrete approaches industry companies have taken for the issue of device sharing. The policy mapping of 18 strategies for $\mathcal{P}_5.\mathcal{RQ}_1$ was discussed in the previous section (and not repeated here in full) identified specific industry approaches to the requirements of the GDPR. For instance, multiple providers opted to handle age-gating with a dynamic approach, where their policy is to forbid usage if the users age is below their national limit, without further specifying it. Or how, for another example, a clear distinction between the need for separate concepts of *identities*, *accounts*, and *profiles* was present in the policies. Further synthesis resulted in a conceptual model that consist of seven deductions, or technical requirements, which would outline a data architecture that enables the identified strategies. These are presented in Figure 7. In the second phase of the study, for $\mathcal{P}_5.\mathcal{RQ}_2$, the previous analysis was evaluated on the concrete level, with a black-box testing of mobile operating systems and applications. Viewing these results for a specific niche use case of device sharing from the general perspective, the lesson is that data protection is complicated and 'the devil is in the details'. The users have various and conflicting needs, which is further confounded with the various and conflicting needs of the different software providing parties in the application ecosystem. The result in $\mathcal{P}_5$ does, for its part, clarify the current status.

As briefly outlined in the results of $\mathcal{P}_1$, one of the research avenues to improve the state-of-the-art for data protection techniques is static analysis. Two different approaches for this, data flow protection and privacy policy generation, taken in $\mathcal{P}_2$ and $\mathcal{P}_3$ are explored in the following section for $\mathcal{D}.\mathcal{RQ}_3$. The implications of the results for the present $\mathcal{D}.\mathcal{RQ}$ is sufficient to summarize as: there is untapped potential in static analysis methods to improve qualitative data protection in present industry practices.

The other perspective into this $\mathcal{D}.\mathcal{RQ}$ is the evaluative lens, that is, how well has the industry responded to the GDPR, which is investigated in $\mathcal{P}_4$. Some of this result was discussed in the previous section, so without repeating that, some further

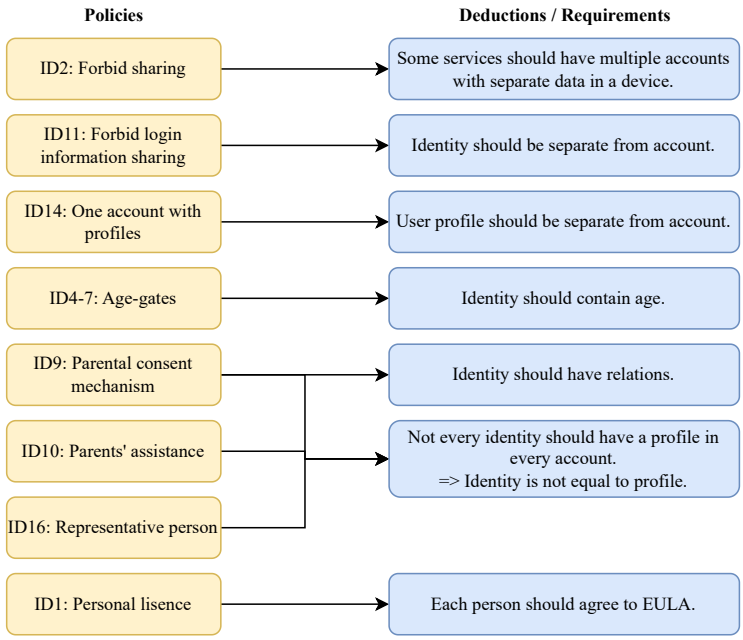

Figure 7. Conceptual model of device sharing resulting from the privacy policy and EULA analysis (from $\mathcal{P}_5$).

points can be made. The dataset is about fines issued, so the analysis cannot judge how often the authorities did find compliance. The amounts of fines distribute mostly between four hundred to two hundred thousand euros, with a few multi-million euro fines present in the long tail. As mentioned, the articles referenced did not have a large difference in the fines imposed, but the most general articles relating to the principles, accountability, and lawfulness were most often referenced. With this result, the following argument is made: the regulators are willing to issue mundane fines in order to push the general level of data protection across the EU upwards, instead of focusing only on major newsworthy breaches. This corroborates the business case for steadily investing in data protection techniques even in SMEs, instead of 'gambling' on not being hit by a major incident.

The answer to this $\mathcal{D}.\mathcal{RQ}$ is the synthesis of multiple metrics ranging from case studies and design science of $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$, to empirical studies of $\mathcal{P}_4$ and $\mathcal{P}_5$ as outlined above. Based on these results, to summarize them in order to provide a direct answer to $\mathcal{D}.\mathcal{RQ}_2$, this dissertation makes the following claim ($\mathcal{C}_2$):

$\mathcal{C}_2$ The current industry practices for software engineering can be improved, in particular the state-of-the-art of the technical measures for data protection required by the GDPR. These improvements are available and reachable with reasonable efforts, as demonstrated by the artifacts designed for

particular cases as a part of this dissertation. Potential avenues for improvement include software architectural techniques, static analysis methods, and ecosystem collaboration. Data protection issues are not simple, and require careful consideration in order to reach solutions that satisfy complex user and data controller needs.

3.2.3 $\mathcal{D}.\mathcal{RQ}_3$: Static analysis of personal data processing

The research question $\mathcal{D}.\mathcal{RQ}_3$ investigates specific techniques for embedding information about personal data processing into source code, using a design science approach. The question statement was *"how can information about personal data processing be embedded in the type systems of software source code, and in what ways this can be applied in software engineering practice?"* The dissertation contributes two different static analysis techniques developed in subsequent phases to meet the issue in $\mathcal{P}_2$ and $\mathcal{P}_3$.

As the first result, in order to define the scope of the analysis, $\mathcal{P}_2.\mathcal{RQ}_1$ presents the results of a literature review of the parallels between static analysis tools positioned for software security and those used for data protection. This serves as the way to organize and focus the efforts and ground the design science to the theoretical background. In the problem characteristics dimension, the result focuses on the *developer* perspective of analysis (as opposed to user, as per the motivation) and the target is *source code* (as opposed to application or ecosystem). The type of analysis chosen is specifically *static analysis*, as the motivation is to reduce the problems of the temporality dimension in software engineering and thus provide the analysis during programming. The level of analysis is a more interesting dimension in terms of focus. In a spectrum from security to data protection, the higher levels of semantic flow and policy analysis are more relevant to data protection. These characteristics are visualized in Fig. 8. This dissertation has the practical software engineering motivation, so the techniques developed target the semantic flow level of software. However, the second part of the research in $\mathcal{P}_3$ intends to bridge the gap from semantic flow into policy, in a particular use case.

The initial methodological contribution to this question is provided in $\mathcal{P}_2.\mathcal{RQ}_2$ with a static analysis technique to control personal data flow within a SOA code base. The motivation is to avoid the proliferation of personal data into those modules of the code base not intended to process personal data, either by developer mistake or misdesign. For instance, a generic logging function could accept an object with personal data as input, but it might not be the intended way of using it. The contribution to increase control over the issue, is to improve the semantics of the type system in question by *denoting the classes that contain personal data* and *denoting the functions which are intended to process personal data*. The analysis, performed by the compiler for example, is then able to warn the developer. This method was

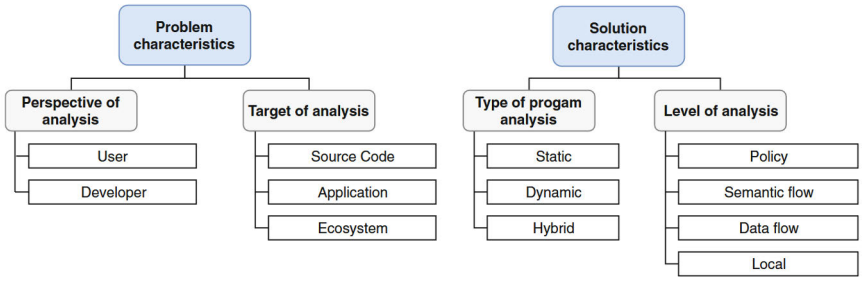

Figure 8. Characteristics of analysis tools for personal data protection (from $\mathcal{P}_2$).

implemented and the concept proved in a case company code base written in the Java language, and the designed tool particularly uses the language annotation constructs to embed the semantics. The annotations can be applied during initial development or on top of any existing code base.

The previous results were extended in the next work package, in $\mathcal{P}_3$, by applying the same semantic improvements into generating data for data to privacy policies based on the structure of the program. The context for this design is, again, a SOA for web services: the service receives a request and processes data to fulfill it. The conceptual formulation for the solution was performed in $\mathcal{P}_3.\mathcal{RQ}_1$. Firstly, an extension that enables composition of privacy policy purposes to the Layered privacy language was designed and proposed. The concept hinges on the notion, that as you break down a personal data processing purpose into its constituent parts, eventually you reach the abstraction level that corresponds to a particular web service request (which can be modeled and composed in turn as e.g. Petri nets). This level of detail goes beyond industry practice, which has been found to be neither specific or explicit enough [41]. With the relation between the SOA and the structure of privacy policies established, the second phase of this study, in $\mathcal{P}_3.\mathcal{RQ}_2$, designed and validated on an industry code base a static analysis tool to generate the corresponding data to these purposes: the personal data processed and data transfers. The implementation was an extension of the tool in $\mathcal{P}_2$ and thus followed the same semantics, and the two kinds of analyses can be deployed side to side.

To summarize these results, the dissertation contributes two novel techniques to improve data protection in software: firstly for the traditional static analysis purpose of reducing programming errors, and secondly to provide specific and accurate description of personal data processing to privacy policies. It is demonstrated that these methods can be implemented and deployed on a proof-of-concept maturity level without a major software engineering effort. The $\mathcal{D}.\mathcal{RQ}_3$ of *"How can information about personal data processing be embedded in the type systems of software source code, and in what ways this can be applied in software engineering practice?"* has two parts to it. The static analysis methods contributed provide two tech-

nical answers to this question, but the larger context of software engineering practice also warrants some arguments. The GDPR requires considering the state-of-the-art for data protection measures when designing software with a risk-appropriate level. As the results demonstrate, static analysis is a practical way of increasing the control over the personal data semantics of a program. This should be considered when making weighty decision such as the choice of programming language. The implication is not that Java used in the study is the optimal language for this dimension (far from it, e.g. entirely new languages are designed to control the semantics [138]), but that the tool belt available for the engineer is different in different software ecosystems. A good static analysis facility is important and facilitates creating these tools. The other argument to be made based on the results is that it is possible to nudge software on the PbD axis [92] from privacy-by-policy *towards* privacy-by-architecture by applying these techniques, or from the static analysis perspective, bringing data from the semantic analysis level to the policy analysis level. This can be performed on existing legacy code bases. In an agile development environment with fast iteration, the up-to-date information of what personal data a program processes is in the program. With this justification, this $\mathcal{D}.\mathcal{RQ}$ concludes in the following claim ($\mathcal{C}_3$):

$\mathcal{C}_3$ It is reasonably practical to embed information about personal data processing into the semantics of traditional software, which can be used to document processing or to reduce accidental errors, and so on. Static analysis tools are within state-of-the-art technical measures for data protection, and the capability to make use of them ought to be considered in design decisions for software architecture. Semantics for personal data processing can be controlled in mundane software without major architecture overhauls such as resorting to academic programming languages. In fast and agile software development, up-to-date information of how personal data is processed is present in the software source code. This ought to be made use of.

3.2.4 $\mathcal{D}.\mathcal{RQ}_4$: Alignment of the ecosystem perspective

The final results of the dissertation in $\mathcal{D}.\mathcal{RQ}_4$ wrap up the dissertation. The question statement was formulated as *"how do data protection requirements align between business objectives, privacy policies, and systems and software architecture?"* The answer to this research questions is a position that is, again, a synthesis of the results in the original research contributions.

Revisiting the motivations of the dissertation, $\mathcal{M}_2$ is concerned with maintaining data protection within the temporality of software development and $\mathcal{M}_4$ maintaining it across competing interests of various parties in an ecosystem. The concluding vision is that there is a path towards a holistic data protection governance schema, with

concrete and independently useful steps to reach it. The ecosystem of software industry, as outlined in Chapter 2 and Fig. 2, iterates on complex socio-legal-technical solutions that govern various software systems. The question statement contains several different relations between these stakeholders.

Considering the relation between data protection requirements and business objectives, as the results of $\mathcal{P}_1$ indicate, eliciting technical requirements from the legal requirements is a nontrivial process (the concept of legal requirement in requirements engineering is not crystal clear either [139]), and that the results also must incorporate business-specific constraints. As discussed in the publication, there are two important takeaways. First is the importance of being able to stay on top of both the ever-evolving regulatory environment (e.g. [140]) in unison with the ever-evolving software business environment. Second is that the technical requirements engineering requires the context of the case at hand. While lawyers can and should be consulted on the regulation, they cannot design the technology. Therefore, it is up to the software architects to be able to bridge this gap from legal requirements to business case specific technical requirements. This conclusion is the first result of this $\mathcal{D}.\mathcal{RQ}$.

The next stage of the argument is based on the static analysis conclusions of $\mathcal{P}_2$ and $\mathcal{P}_3$, with some more generalization. As argued in $\mathcal{C}_3$, the techniques designed in the original contributions demonstrate both that it is possible to find improvements in theory and, more importantly, that applying them is reasonably practical in the context of the case systems. The results of $\mathcal{P}_3.\mathcal{RQ}_1$ show that reasoning on the composition of dependencies as enabled by SOA empowers bridging an analysis from the semantic flow level into the policy level. Is SOA uniquely suitable for this is not in the scope of the argument, which is that the architecture should enable such reasoning. Furthermore, though it is not explicitly studied in this dissertation, a reasonable assumption based on industry experience is that there is a difference in the quality of static analysis facilities for different programming languages, frameworks, et cetera. Even if there were no fundamental theoretical limit on the analysis one can program, the facilities have practical consequences on the approachability and affordability for actual software engineering. The conclusion is that ordinary software without special privacy-by-architecture patterns can support advanced techniques to reinforce data protection in the privacy-by-policy model, when its architecture enables reasoning on the dependency tree.

Should a hypothetical company follow along the argument with corresponding implementation, they would at this point have the business capability to elicit company business objective specific technical requirements to meet regulation, and fully static analysis capable software architecture. As discussed in $\mathcal{P}_3$, the next step is to make use of this information in data and process governance. The generalized version of the results is the demonstration of two instances of specific governance patterns that are enabled by the static analysis. The present use case of generating

up-to-date specific and explicit data to privacy policies from software architecture could be used to directly empower data subjects. There is plenty of potential in applying the pattern to other documentation, such as API descriptions, linking the result to the semantic web body of work and so on.

The final inference to be made in relation to $\mathcal{D}.\mathcal{RQ}_4$ are the results of $\mathcal{P}_5$, in the relation of the alignment of the software business objectives to the other considered elements. The results of the study serve as an example how the ecosystem has managed the cycles of *regulatory technology transfer* and *software service-usage-enforcement feedback activities* for the specific data protection issues around device sharing. As discussed in $\mathcal{P}_5$, the industry actors have employed varied data protection strategies, and both common elements and diverging implementations were identified. The divergence is present in the policy level as well, requiring interpretation of terms and conditions to parse. The result for the purposes of this argument is twofold. Firstly, the current situation is not optimal for the data subjects, as even identifying the device sharing rules was a research project, nor for the different ecosystem actors, since the applications and services do not interoperate. Secondly, the common elements present in the policies represent an opportunity to improve the situation with, for instance, the methods proposed in this dissertation. The final step proposed here is a conjecture reached based on the previous results: the internet services ecosystem could further develop common open data models and interoperability protocols for personal data processing, and any notoriously difficult standardization coordination could be reduced by the previous improvements in, for instance, available privacy policy data.

Revisiting the definition of the $\mathcal{D}.\mathcal{RQ}_4$, the alignment of the different facets of data protection requirements, business objectives, privacy policies and systems and software architecture were studied. They were found interconnected and interdependent in practice, as expected. As the discipline of the dissertation is software engineering, the perspective emphasizes the role of architecture in the improvement of the different domains. Data protection requirements and business objectives are integrated in technical requirements, and software techniques are the source for privacy policy data governance. Data protection, business objectives and privacy policies are aligned with an ecosystem view onto the issue with an interoperability motivation. These results incorporate into the final claim made in this dissertation ($\mathcal{C}_4$):

$\mathcal{C}_4$ The following playbook is a path with potential to improved data protection in ordinary transaction processing software, where the intermediary steps each independently provide practical value and can be completed without major industry co-operation:

Step 1: Software architects ought to ensure the ability to translate legal requirements into actionable software requirements thereby bridging the gap between legal frameworks and system design.

Step 2: Architect software in a way that enables reasoning about the network of intra and inter system dependencies, for example SOA and static analysis facilities.

Step 3: Apply static analysis to gain a complete model of personal data processing semantics in the software stack.

Step 4: Apply the above model in privacy policies and service descriptions to empower data subjects and ecosystem actors.

Step 5: Apply the above in industry collaboration to develop common data models and personal data features across platform providers and users.

This claim, in part, reinforces the notion that developers must embrace data protection, and outlines a position on global data protection governance. With this, the results of this dissertation are concluded; the next section discusses them further.

4 Discussion

To wrap up this dissertation, this chapter discusses the main limitations of the research conducted and the results reached in Section 4.1, the implications of the work to both academia and practice in Section 4.2, and potential avenues for future research in Section 4.3.

4.1 Limitations

This section outlines the limitations of the dissertation. The main focus is on the dissertation level, and the limitations of the individual contributions are further elaborated in the publications themselves. After some notes on the research setting in general, the rest of this discussion is organized around the research questions $\mathcal{D}.\mathcal{RQ}_1$ through $\mathcal{D}.\mathcal{RQ}_4$ and their answering claims $\mathcal{C}_1$ through $\mathcal{C}_4$ made in the previous chapter.

The dissertation strives for two goals, as denoted $\mathcal{G}_1$ and $\mathcal{G}_2$ previously. Both are formulated as aiming for *improvements* for different dimensions, summarized as to the understanding of the GDPR and to the state-of-the-art of data protection methods. These goals are met with the four $\mathcal{D}.\mathcal{RQ}$s on the operational level of the work. The original research contributions provide metrics to the $\mathcal{D}.\mathcal{RQ}$s which, in turn, are argued to further the goals. As the goals are generic, it is straightforward to argue that some improvement was made, as evidenced by the acceptance of the publications, as long as the contributions are even directionally correct. The main limitation in this regard is that the *extent* of the improvement is not measured in this work.

These improvements sought can result in an effect on multiple different areas of influence, with different ways to measure it, ranging from academia to industry to regulations, as described in Section 2.2.1 previously. Some of this effect occurs after time with the dissemination of the research, including this dissertation, and academic impact has established indicators including citation counts. The research has been committed over a number of years and, as such, some of these metrics are available for the original publications. As technology transfer was the overarching setting of the dissertation, that warrants some discussion. Technology transfer to the industry can be viewed on different levels ranging from global impact, to companies, to single practitioners. So far the research appears not to have notable impacts globally. The

tools designed are published as open source, but are not polished or marketed, and have no direct traction as such. The technology transfer impact is, so far, limited to the author gaining knowledge from academia and bringing it to individual contracted organizations, among other dissemination.

As to the validity of the research, some notes are warranted. The results are summarized in the four claims, which, in turn, were posited as answers to four $\mathcal{D}.\mathcal{RQ}$s, formulated on a general level. The main possible objections to the design of the $\mathcal{D}.\mathcal{RQ}$s and the treatment are 1) are the questions conclusively answerable, and 2) do the results of the original contributions answer them? Both of these are legitimate concerns, and possible risks of the dissertation research design. While the approach does follow common formulation patterns for design science research [141], the cause for this limitation is the problem-solving approach, which does not *prove* the answers. The dissertation combines case studies, design science, and requirements engineering, in which problem-solving is an objective and conclusive general answers are difficult [141]. There are rigorous knowledge-generating parts in the dissertation, and other parts demonstrate novel ways to solve problems – both are sound in their own ways. With these limitations of the overall approach in mind, the author does argue that the $\mathcal{D}.\mathcal{RQ}$s were treated to satisfaction, with a justified answer to each, though further question-specific limitations apply. Let us then discuss these specific claims and their limitations.

The topic of the first $\mathcal{D}.\mathcal{RQ}_1$ is the requirements of the GDPR and its implications to software engineering, and topic of $\mathcal{D}.\mathcal{RQ}_2$ the industry meeting these demands. These were approached in different $\mathcal{P}.\mathcal{RQ}$s through requirements engineering, case studies, design science, literature review, and archival research. Internal validity for the specific metrics is controlled with various means, as explained in the original publications in further detail, including inter-rater agreement and validation in the industry for the design artifacts. Ultimately, none of the results have been tested in court or with data protection authorities, as far as the author is aware, which is a limitation. For example, as a hypothetical way the authorities could corroborate the results, an organization being sued for a data protection issue could present the use of the methods in this dissertation as technical and organizational protection measures (where applicable), and the court accept that as evidence of them not being negligent in the case. External validity of the results and the claims have further limitations. While the results based on archival data are repeatable, the case studies are not. However, applying the patterns in other contexts would provide corroboration or counter-examples. While, arguably, each of the contributions have some results that generalize in part, they are far from universal. For instance, $\mathcal{P}_1$ is focused on SMEs in particular, and emphasizes that the business context is an input to the requirements engineering process. While SMEs as a category does represent 99% of businesses in the EU [142], it is not a far-fetched conjecture to say the large multinational corporations have the most impact on personal data processing. The data set

of $\mathcal{P}_4$ is skewed geographically and the results could change with a more mature set and further enforcement actions. While the research in $\mathcal{P}_5$ identifies both issues and common patterns of solutions from the industry, they are limited to a small niche of the overall personal data processing space.

The third question $\mathcal{D}.\mathcal{RQ}_3$ contains the main technological contributions made in this dissertation, the two static analysis techniques for reducing data misuse and generating privacy policy data. These, too, are design science contributions, and the validity should be looked in that light. Internal validity is justified by testing in industry case studies, though these are small-scale proofs-of-concepts. External validity is not entirely guaranteed. The tools for the techniques being open source allows for validating against other scenarios or modifying the tools to be suitable for different cases. The expectation is that the techniques are valid, but it is not claimed that these are the only or optimal ways to have a similar effect. The validation is limited to demonstrating the techniques are implementable, and the effect of their usage in software engineering is not measured. These are the main limitation of the contributions. The claim this dissertation makes in $\mathcal{C}_3$ has several propositions. The majority of them are formulated as being correct by demonstration of an example (for instance, "it is reasonably practical to embed information about personal data processing into the semantics of traditional software"). The claim makes two abductive leaps, to argue that static information about personal data processing in source code should be made use of, and these are not proven, only justified.

The final part of the dissertation in $\mathcal{D}.\mathcal{RQ}_4$ attempts to make a synthesis of the topic by combination of the previous results. The outcome is a five-step playbook to that is claimed to be a path to improved data protection in ordinary software, with independently worthwhile steps. These have been reached with abductive reasoning, the justification of which is elaborated in Section 3.2. These are not posited to be universal, but come with caveats and limitations. Supposedly, the proposition generalizes better to those circumstances that are similar to the ones in the case studies of the dissertation, than wholly different situations. Furthermore, the empirical effect of applying the entire set of recommendations is not measured. For instance, the final step is mostly conjecture, as the dissertation merely highlights the issue, and does not evaluate a remedy.

To summarize the general theme of the limitations of this dissertation, it is a twofold combination: the studies were exploratory research on emerging recent issues at the time of research, and the approach taken is problem-solving rather than theory-building. This results in weaker data sets and conclusions, but also timely answers and practical insights.

4.2 Implications

Some further discussion on the implications of the results of this dissertation is warranted and follows in this section. This is in addition to the discussion in the original publications as well.

The dissertation corroborates the notion that there are privacy engineering techniques that are reasonable to implement and develop – in ordinary software without major overhauls. As discussed, there is a gap between privacy engineering research and industry practice [98]. The strategy of this dissertation is to further contribute to techniques that apply for ordinary software, where the quality of privacy-by-policy systems could be improved and nudge them with applicable tools on the spectrum towards privacy-by-architecture. The argument is not that developing a privacy-by-architecture solution is more difficult per se, but that organizations (especially SMEs) might not have the resources to entirely redesign systems and that small changes are easier to accept. The novelty of the results in this dissertation is that the tools can be applied on top of existing legacy code-bases. If this line of development does not bear fruit as industry improvements, though the present dissertation is but a drop in the ocean and alone does not warrant the conclusion, it serves as evidence that the state-of-the-art and practice gap is a matter of *will* rather than possibility. That, in turn, would serve as feedback to the regulatory technology transfer activity described in Section 2.2.1.

The vision of static analysis information governance outlined in claims $\mathcal{C}_3$ and $\mathcal{C}_4$ goes beyond the specific result use case of generating data for privacy policies. The fully realized ideal is the complete static data and process model, of which the as-is version is *current* as it is generated from the ever evolving source code itself. This can be applied in business development, architecture planning, audits, and so on. The recommendation could be further generalized from the structure of software itself to, for example, infrastructure-as-code patterns or configuration management, though this has not been explored in this dissertation. The implication of this is that there is still vast potential for expanding and combining these data sources. Even if, by themselves, these techniques are standard operating procedure in part, the entire combination is far from best practice status at the moment. Ultimately, a software engineering organization could gather the structure of their entire data processing stack into one static model – one that is updated automatically on changes to the stack.

These developments in this dissertation contribute to the semantic web conversation. The concept that online services statically describe their semantics, in order to enable algorithmic usage, is decades old, though it has been argued not to have materialized in practice [89]. It could be speculated that this effort might get revitalized in the near future with new developments in user agents. The data generated by the tools of this dissertation can be combined with OpenAPI-specifications for personal

data semantics [143], for instance, to contribute into the semantic web. There is no strict connection: you can have semantic web without personal data processing semantics or not, but should there be a resurgence, it would be better for transparency and our rights to incorporate privacy semantics as well.

One of the main implications of the dissertation is that as everything in regulation, businesses and software are always changing (and feeding back to each other), and as the mapping from legal requirements to technical requirements requires business specific constraints, software architects must stay on top of implementing regulation. At the time of writing, the GDPR has been proposed to be reopened for changes [144; 145]. The proposed changes would reduce record keeping obligations for SMEs and mid-cap companies in certain circumstances, but determining whether the conditions apply is arguable equally laborious [146] – labor which is reduced by having a model of your data processing activities. Furthermore, new legislation such as the Cyber Resilience Act has been enacted, with new requirements [140]. All this corroborates the claim $\mathcal{C}_4$ that the capability to adjust to change is necessary, and with a further conjecture that having an up-to-date static model of personal data processing makes it easier to reason about the impact of changing regulation to a particular software system.

The perspective of the dissertation is chiefly from the software engineering side of a software developing organization. As discussed in the introduction, much responsibility is being placed on the cross-functional agile teams in that development paradigm. Well, the results of this dissertation imply even more responsibility to the agile teams (and thus power): maintaining the personal data processing model that feeds directly into privacy policies, for instance. Policies and contracts are often, at least in larger organizations, without arguing specific numbers, the responsibility of dedicated teams of business professionals including data protection lawyers. The proposals of this dissertation do represent a shift to the model, but do not necessarily imply as fundamental a change of power as it might seem on the surface; rather, it fixes the flow of information. In simplified terms, under the privacy-by-policy model a business process owner dictates a policy, which is used as requirements for software engineering, and is published as a privacy policy. This dissertation argues that the business policy would still provide requirements for software engineering, but that the privacy policy is maintained based on the actual software ensuring reality matches the text.

The fourth motivation for the dissertation was examining the data protection rights in various ecosystems. Essentially, this cannot be escaped and viewed from the outside: the present text fits into the academia-industry-government -ecosystem in itself. The research approach is a demonstration of the technology transfer process with the three design science cycles and some additional empirical observation. This dissertation both identifies issues ($\mathcal{P}_1$, $\mathcal{P}_4$, $\mathcal{P}_5$) and contributes concrete solutions to the industry feedback loop (all publications). Most of the contributions take the

perspective of an individual software engineering organization (aside from $\mathcal{P}_5$), but the dissertation does have aspirations of affecting the ecosystem level as well. This is expressed in $\mathcal{C}_4$ of the results. The conjecture here is that ecosystem wide data protection improvements can be eased in by providing best practices for individual organizations – the standardization will follow. Developing, collecting, and disseminating best practices also changes the GDPR state-of-the-art protection measures evaluation and the risk calculus.

4.3 Future work

Much of the work in this dissertation is exploratory in nature, which has the value of responding quickly to emerging topics. There is plenty of research space in both maturing these lines of exploration, though, and many new opportunities uncovered in addition. Here are are some pointers the present author is interested in and engaged in some.

Maturing

The requirements analysis and case study architectures presented in $\mathcal{P}_1$ could be validated in more general contexts and with further depth into specific use cases. Revisiting the requirements periodically as both the regulation and the data protection environment evolves could prove fruitful. The methods published in $\mathcal{P}_2$ and $\mathcal{P}_3$ could be also corroborated with further validation in larger contexts, and the tools developed into marketable operationalized products with support for different programming languages and so on. The effect of the proposed methods could be measured from various perspectives. The results of $\mathcal{P}_4$ are based on a limited data set of early GDPR enforcement data, and thus the understanding of this space definitely would benefit from repeated analysis and consolidation of the data sources. While the policy analysis in $\mathcal{P}_5$ is solid, the designed solutions were not validated as a whole, and the implications of the challenges identified in the platform ecosystems for policy could be developed further.

New opportunities

Even at the time of writing, years after the initial publication of $\mathcal{P}_1$, there are no established technical standards for GDPR compliance or certification processes, at least commonly available (not to claim there has been no maturing). This is in contrast to the information security domain, for instance, where multiple different criteria and standards are available. Beyond internal data controller compliance, the space for controller inter-operability is also underdeveloped. This could be addressed, in part, by making use of the results of this dissertation. The regulation evolves, and beyond revisiting the analysis of $\mathcal{P}_1$, there is also space for cross-examination of the technical requirements of different regulations (e.g. the Cyber Resilience Act [140]) – and the requirements analysis undertaken for the GDPR is useful as a base for other compatible regulation. Both within the EU and globally. The static analysis tools developed in $\mathcal{P}_2$ and $\mathcal{P}_3$ represent certain use cases of a general methodology, which can both be developed and integrated further, but also extended to new data protection use cases. For instance, this dissertation has not put its focus on the data protection impact assessment process, and the ability of static analysis to contribute to that. Some

empirical questions include: Is SOA uniquely suited for data flow analysis? How do the results presented affect the software engineering process? Finally, the analysis of $\mathcal{P}_5$ can be used as an example to investigate other "edge cases" of data protection implementation.

List of References

[1] Kalle Rindell. *Development of Secure Software: Rationale, Standards and Practices*. University of Turku, 2019.

[2] P Borque and R Fairley. Guide to the software engineering body of knowledge version 3.0. *IEEE Computer Society Staff*, 2014.

[3] Mireille Hildebrandt. *Law for computer scientists and other folk*. Oxford University Press, 2020.

[4] Daniel J Solove. Conceptualizing privacy. *California Law Review*, 90:1087–1156, 2002.

[5] Seda Gürses and Jose M. del Alamo. Privacy Engineering: Shaping an Emerging Field of Research and Practice. *IEEE Security & Privacy*, 14(2):40–46, 2016.

[6] Juergen Musil, Angelika Musil, Danny Weyns, and Stefan Biffl. Society-Level Software Governance: A Challenging Scenario. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, pages 307–308, 2020.

[7] David Budgen. *Software design*. Pearson Education, 2003.

[8] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, Anders Wesslén, et al. *Experimentation in software engineering*, volume 236. Springer, 2012.

[9] Victor R Basili. Quantitative evaluation of software methodology. Technical report, University of Maryland, 1985.

[10] Victor Basili, Gianluigi Caldiera, and H Dieter Rombach. The goal question metric approach. *Encyclopedia of software engineering*, pages 528–532, 1994.

[11] European Union. Charter of Fundamental Rights of the European Union. *Official Journal of the European Union C83*, 53:380, 2010.

[12] Gloria González Fuster. *The emergence of personal data protection as a fundamental right of the EU*, volume 16. Springer Science & Business, 2014.

[13] European Parliament and the Council of the European Union. 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), May 2016.

[14] David Vincent. *Privacy: a short history*. John Wiley & Sons, 2016.

[15] UN General Assembly. International Covenant on Civil and Political Rights. *Treaty Series*, 1966. Adopted and opened for signature, ratification and accession by General Assembly resolution 2200 A (XXI) of 16 Dec. 1966.

[16] Sofija Voronova and Anna Nichols. Understanding EU data protection policy. *European Parliamentary Research Service*, 2020.

[17] European Commission. Communication from the commission to the european parliament, the council, the european economic and social committee and the committee of the regions safeguarding privacy in a connected world a european data protection framework for the 21st century, com/2012/09, 2012.

[18] Christina Tikkinen-Piri, Anna Rohunen, and Jouni Markkula. EU General Data Protection Regulation: Changes and implications for personal data collecting companies. *Computer Law & Security Review*, 34(1):134–153, 2018.

[19] European Parliament and the Council of the European Union. Regulation (EU) 2018/1725 of the European Parliament and of the Council of 23 October 2018 on the protection of natural

persons with regard to the processing of personal data by the Union institutions, bodies, offices and agencies and on the free movement of such data, and repealing Regulation (EC) No 45/2001 and Decision No 1247/2002/EC (Text with EEA relevance.), November 2018.

[20] European Parliament and the Council of the European Union. Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence and amending Regulations (EC) No 300/2008, (EU) No 167/2013, (EU) No 168/2013, (EU) 2018/858, (EU) 2018/1139 and (EU) 2019/2144 and Directives 2014/90/EU, (EU) 2016/797 and (EU) 2020/1828 (Artificial Intelligence Act) (Text with EEA relevance), 2024.

[21] European Commission. Proposal for a regulation of the european parliament and of the council on the european health data space, 2022.

[22] Peter Blume. The inherent contradictions in data protection law. *International Data Privacy Law*, 2(1):26–34, 2012.

[23] Bert-Jaap Koops. The trouble with the European data protection law. *International Data Privacy Law*, 4(4):250–252, 2014.

[24] J.C. Buitelaar. Privacy: Back to the roots. *German Law Journal*, 13(3):171–202, 2012.

[25] Antoinette Rouvroy and Yves Poullet. The right to informational self-determination and the value of self-development: Reassessing the importance of privacy for democracy. In *Reinventing data protection?*, pages 45–76. Springer, 2009.

[26] Florent Thouvenin. Informational self-determination: a convincing rationale for data protection law? *Journal of Intellectual Property, Information Technology, and Electronic Commerce Law*, 12:246, 2021.

[27] Michèle Finck and Frank Pallas. They who must not be identified—distinguishing personal from non-personal data under the GDPR. *International Data Privacy Law*, 10(1):11–36, 2020.

[28] Supriya Adhatarao, Cédric Lauradoux, and Cristiana Santos. Why Is My IP Address Processed? No Data For Accountless Users. In *Privacy Symposium: Data Protection Law International Convergence and Compliance with Innovative Technologies*, pages 231–250. Springer, 2022.

[29] Jukka Ruohonen and Sini Mickelsson. Reflections on the data governance act. *Digital Society*, 2(1):10, 2023.

[30] Piero A Bonatti, Sabrina Kirrane, Iliana M Petrova, and Luigi Sauro. Machine understandable policies and GDPR compliance checking. *KI-Künstliche Intelligenz*, 34:303–315, 2020.

[31] Patrick Gage Kelley, Lucian Cesca, Joanna Bresee, and Lorrie Faith Cranor. Standardizing privacy notices: an online study of the nutrition label approach. In *Proceedings of the SIGCHI Conference on Human factors in Computing Systems*, pages 1573–1582, 2010.

[32] Mike Hintze. In defense of the long privacy statement. *Maryland Law Review*, 76:1044, 2016.

[33] Aleecia M McDonald, Robert W Reeder, Patrick Gage Kelley, and Lorrie Faith Cranor. A comparative study of online privacy policies and formats. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 37–55. Springer, 2009.

[34] Robert W. Reeder, Patrick Gage Kelley, Aleecia M. McDonald, and Lorrie Faith Cranor. A user study of the expandable grid applied to p3p privacy policy visualization. In *Proceedings of the 7th ACM Workshop on Privacy in the Electronic Society*, WPES '08, page 45–54, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605582894.

[35] Norman Sadeh, Alessandro Acquisti, Travis D Breaux, Lorrie Faith Cranor, Aleecia M McDonald, Joel R Reidenberg, Noah A Smith, Fei Liu, N Cameron Russell, Florian Schaub, et al. The usable privacy policy project. In *Technical report, Technical Report, CMU-ISR-13-119*. Carnegie Mellon University, 2013.

[36] Jana Korunovska, Bernadette Kamleitner, and Sarah Spiekermann. The challenges and impact of privacy policy comprehension. In *Proceedings of the 28th European Conference on Information Systems*, ECIS '20, 2020.

[37] Christine Utz, Martin Degeling, Sascha Fahl, Florian Schaub, and Thorsten Holz. (Un) informed consent: Studying GDPR consent notices in the field. In *Proceedings of the 2019 acm sigsac conference on computer and communications security*, pages 973–990, 2019.

[38] Armin Gerl, Nadia Bennani, Harald Kosch, and Lionel Brunie. LPL, towards a GDPR-compliant privacy language: formal definition and usage. *Transactions on Large-Scale Data-and Knowledge-Centered Systems XXXVII*, pages 41–80, 2018.

[39] Mina Deng, Kim Wuyts, Riccardo Scandariato, Bart Preneel, and Wouter Joosen. A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements. *Requirements Engineering*, 16(1):3–32, 2011.

[40] Jose M Del Alamo, Danny S Guaman, Boni García, and Ana Diez. A systematic mapping study on automated analysis of privacy policies. *Computing*, 104(9):2053–2076, 2022.

[41] Célestin Matte, Cristiana Santos, and Nataliia Bielova. Purposes in IAB Europe's TCF: which legal basis and how are they used by advertisers? In *Privacy Technologies and Policy: 8th Annual Privacy Forum, APF 2020, Lisbon, Portugal, October 22–23, 2020, Proceedings 8*, pages 163–185. Springer, 2020.

[42] Damiano Torre, Sallam Abualhaija, Mehrdad Sabetzadeh, Lionel Briand, Katrien Baetens, Peter Goes, and Sylvie Forastier. An ai-assisted approach for checking the completeness of privacy policies against gdpr. In *2020 IEEE 28th International Requirements Engineering Conference (RE)*, pages 136–146. IEEE, 2020.

[43] Martin Degeling, Christine Utz, Christopher Lentzsch, Henry Hosseini, Florian Schaub, and Thorsten Holz. We value your privacy... now take some cookies: Measuring the GDPR's impact on web privacy. In *Network and Distributed System Security Symposium 2019*. The Internet Society, 2019.

[44] Timi Heino, Robin Carlsson, Sampsa Rauti, and Ville Leppänen. Assessing discrepancies between network traffic and privacy policies of public sector web services. In *Proceedings of the 17th international conference on availability, reliability and security*, pages 1–6, 2022.

[45] Danny S. Guamán, David Rodriguez, Jose M. del Alamo, and Jose Such. Automated GDPR compliance assessment for cross-border personal data transfers in android applications. *Computers & Security*, 130:103262, 2023.

[46] Kenneth A Bamberger and Deirdre K Mulligan. New governance, chief privacy officers, and the corporate management of information privacy in the United States: An initial inquiry. *Law & Policy*, 33(4):477–508, 2011.

[47] David G Hill. *Data protection: Governance, risk management, and compliance*. CRC Press, 2016.

[48] Sarah Spiekermann and Lorrie Faith Cranor. Engineering privacy. *IEEE Transactions on Software Engineering*, 35(1):67–82, 2009.

[49] Johannes Holvitie. *Technical Debt in Software Development: Examining Premises and Overcoming Implementation for Efficient Management*. PhD thesis, University of Turku, 2017.

[50] Edsger W Dijkstra et al. On the cruelty of really teaching computing science. *Communications of the ACM*, 32(12):1398–1404, 1989.

[51] Alan M. Davis, Edward H. Bersoff, and Edward R. Comer. A strategy for comparing alternative software development life cycle models. *IEEE Transactions on Software Engineering*, 14(10): 1453–1461, 1988.

[52] Martin Fowler, Jim Highsmith, et al. The agile manifesto. *Software development*, 9(8):28–35, 2001.

[53] Ken Schwaber and Jeff Sutherland. The Scrum guide. *Scrum Alliance*, 21(1):1–38, 2011.

[54] Kent Beck and Martin Fowler. *Planning extreme programming*. Addison-Wesley Professional, 2001.

[55] Jonas Eckhardt, Andreas Vogelsang, and Daniel Méndez Fernández. Are "Non-functional" Requirements really Non-functional? An Investigation of Non-functional Requirements in Practice. In *Proceedings of the 38th international conference on software engineering*, pages 832–842, 2016.

[56] Sandun Dasanayake, Sanja Aaramaa, Jouni Markkula, and Markku Oivo. Impact of requirements volatility on software architecture: How do software teams keep up with ever-changing requirements? *Journal of software: evolution and process*, 31(6):e2160, 2019.

[57] Frank Bott, Allison Coleman, and Diane Rowland. *Professional issues in software engineering*. CRC Press, 2000.

[58] Bishop Matt. Computer security: art and science, 2002.

[59] Nancy R Mead, Julia H Allen, Sean Barnum, Robert J Ellison, and Gary R McGraw. *Software security engineering: a guide for project managers*. Addison-Wesley Professional, 2004.

[60] Barry W. Boehm. Verifying and validating software requirements and design specifications. *IEEE software*, 1(1):75, 1984.

[61] Timo OA Lehtinen, Mika V Mäntylä, Jari Vanhanen, Juha Itkonen, and Casper Lassenius. Perceived causes of software project failures–an analysis of their relationships. *Information and Software Technology*, 56(6):623–643, 2014.

[62] Adam A Porter, Lawrence G Votta, and Victor R Basili. Comparing detection methods for software requirements inspections: A replicated experiment. *IEEE Transactions on software Engineering*, 21(6):563–575, 1995.

[63] Julian Frattini, Lloyd Montgomery, Jannik Fischbach, Daniel Mendez, Davide Fucci, and Michael Unterkalmsteiner. Requirements quality research: a harmonized theory, evaluation, and roadmap. *Requirements engineering*, 28(4):507–520, 2023.

[64] Klaus Julisch, Christophe Suter, Thomas Woitalla, and Olaf Zimmermann. Compliance by design–Bridging the chasm between auditors and IT architects. *Computers & Security*, 30(6-7):410–426, 2011.

[65] Guenther Ruhe, Maleknaz Nayebi, and Christof Ebert. The vision: Requirements engineering in society. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 478–479. IEEE, 2017.

[66] Walid Maalej, Maleknaz Nayebi, Timo Johann, and Guenther Ruhe. Toward data-driven requirements engineering. *IEEE software*, 33(1):48–54, 2015.

[67] Silvia Ingolfo, Alberto Siena, John Mylopoulos, Angelo Susi, and Anna Perini. Arguing regulatory compliance of software requirements. *Data & Knowledge Engineering*, 87:279–296, 2013.

[68] Robert Muthuri, Guido Boella, Joris Hulstijn, and Llio Humphreys. Argumentation-based legal requirements engineering: the role of legal interpretation in requirements acquisition. In *2016 IEEE 24th International Requirements Engineering Conference Workshops (REW)*, pages 249–258. IEEE, 2016.

[69] Claudia Negri-Ribalta, Marius Lombard-Platet, and Camille Salinesi. Understanding the GDPR from a requirements engineering perspective—a systematic mapping study on regulatory data protection requirements. *Requirements Engineering*, pages 1–27, 2024.

[70] Paul Ralph. The two paradigms of software development research. *Science of Computer Programming*, 156:68–89, 2018.

[71] D Méndez Fernández, M-T Christiansson, et al. Naming the pain in requirements engineering: Contemporary problems, causes, and effects in practice. *Empirical software engineering*, 22(5): 2298–2338, 2017.

[72] Dewayne E Perry and Alexander L Wolf. Foundations for the study of software architecture. *ACM SIGSOFT Software engineering notes*, 17(4):40–52, 1992.

[73] Grady Booch. The accidental architecture. *IEEE Software*, 23(3):9–11, 2006.

[74] Philippe B Kruchten. *The 4+1 view model of architecture*, volume 12. IEEE, 1995.

[75] Andrew Josey, Marc Lankhorst, Iver Band, Henk Jonkers, and Dick Quartel. An introduction to the archimate® 3.0 specification. *White Paper from The Open Group*, 2016.

[76] The Open Group. ArchiMate® 3.1 Specification. Standard, The Open Group, 2019.

[77] Svyatoslav Kotusev. Enterprise Architecture on a Single Page. *British Computer Society (BCS)*, 2017.

[78] Melvin E Conway. How do committees invent. *Datamation*, 14(4):28–31, 1968.

[79] RC Martin. *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Pearson Education Inc, 2018.

[80] Robert C Martin. Design principles and design patterns. *Object Mentor*, 1(34):597, 2000.

[81] John A Zachman. A framework for information systems architecture. *IBM systems journal*, 26 (3):276–292, 1987.
[82] Marc Lankhorst. *Enterprise Architecture at Work: Modelling, Communication and Analysis*. Springer, 2009.
[83] Van Haren. *TOGAF Version 9.1*. Van Haren Publishing, 2011.
[84] Brian Chess and Jacob West. *Secure programming with static analysis*. Pearson Education, 2007.
[85] Pramod Mathew Jacob and M. Prasanna. A Comparative analysis on Black box testing strategies. In *2016 International Conference on Information Science (ICIS)*, pages 1–6, 2016.
[86] WeiTek Tsai, XiaoYing Bai, and Yu Huang. Software-as-a-service (SaaS): perspectives and challenges. *Science China Information Sciences*, 57:1–15, 2014.
[87] Matthias Klusch, Patrick Kapahnke, Stefan Schulte, Freddy Lecue, and Abraham Bernstein. Semantic web service search: a brief survey. *KI-Künstliche Intelligenz*, 30:139–147, 2016.
[88] Angel Lagares Lemos, Florian Daniel, and Boualem Benatallah. Web service composition: a survey of techniques and tools. *ACM Computing Surveys (CSUR)*, 48(3):1–41, 2015.
[89] Nigel Shadbolt, Tim Berners-Lee, and Wendy Hall. The semantic web revisited. *IEEE intelligent systems*, 21(3):96–101, 2006.
[90] Jesús García-Galán, Liliana Pasquale, George Grispos, and Bashar Nuseibeh. Towards adaptive compliance. In *Proceedings of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 108–114, 2016.
[91] Daniel J. Solove. A taxonomy of privacy. *University of Pennsylvania Law Review*, 154:477, 2005-2006.
[92] Sarah Spiekermann and Lorrie Faith Cranor. Engineering privacy. *IEEE Transactions on Software Engineering*, 35(1):67–82, 2009.
[93] Gloria González, Rosamunde Van Brakel, and Paul De Hert. *Research handbook on privacy and data protection law: values, norms and global politics*. Edward Elgar Publishing, 2022.
[94] Ann Cavoukian et al. Privacy by design: The 7 foundational principles. *Information and privacy commissioner of Ontario, Canada*, 5:12, 2009.
[95] Seda Gürses, Carmela Troncoso, and Claudia Diaz. Engineering privacy by design. *Computers, Privacy & Data Protection*, 14(3):25, 2011.
[96] Jeroen van Rest, Daniel Boonstra, Maarten Everts, Martin van Rijn, and Ron van Paassen. Designing privacy-by-design. In *Privacy Technologies and Policy: First Annual Privacy Forum, APF 2012, Limassol, Cyprus, October 10-11, 2012, Revised Selected Papers 1*, pages 55–72. Springer, 2014.
[97] Jukka Ruohonen. "By design" principles considered harmful. *Internet Policy Review*, 2025.
[98] Luke Stark, Jen King, Xinru Page, Airi Lampinen, Jessica Vitak, Pamela Wisniewski, Tara Whalen, and Nathaniel Good. Bridging the gap between privacy by design and privacy in practice. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pages 3415–3422, 2016.
[99] George Danezis, Josep Domingo-Ferrer, Marit Hansen, Jaap-Henk Hoepman, Daniel Métayer, Rodica Tirtea, and Stefan Schiffner. *Privacy and Data Protection by Design - from Policy to Engineering*. European Union Agency for Network and Information Security (ENISA), 12 2014.
[100] Spyros Kokolakis. Privacy attitudes and privacy behaviour: A review of current research on the privacy paradox phenomenon. *Computers & security*, 64:122–134, 2017.
[101] Idris Adjerid, Eyal Peer, and Alessandro Acquisti. Beyond the Privacy Paradox: Objective Versus Relative Risk in Privacy Decision Making. *MIS Quarterly*, 42(2):pp. 465–488, 2018. ISSN 02767783, 21629730.
[102] Ira S Rubinstein and Nathaniel Good. Privacy by design: A counterfactual analysis of Google and Facebook privacy incidents. *Berkeley Tech. LJ*, 28:1333, 2013.
[103] Irit Hadar, Tomer Hasson, Oshrat Ayalon, Eran Toch, Michael Birnhack, Sofia Sherman, and Arod Balissa. Privacy by designers: software developers' privacy mindset. *Empirical Software Engineering*, 23:259–289, 2018.

[104] Awanthika Senarath and Nalin A. G. Arachchilage. Why developers cannot embed privacy into software systems? An empirical investigation. In *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018*, EASE '18, page 211–216, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450364034.

[105] Laurens Sion, Pierre Dewitte, Dimitri Van Landuyt, Kim Wuyts, Ivo Emanuilov, Peggy Valcke, and Wouter Joosen. An architectural view for data protection by design. In *2019 IEEE International Conference on Software Architecture (ICSA)*, pages 11–20. IEEE, 2019.

[106] Vanessa Ayala-Rivera, A Omar Portillo-Dominguez, and Liliana Pasquale. GDPR compliance via software evolution: Weaving security controls in software design. *Journal of Systems and Software*, 216:112144, 2024.

[107] Pierre Dewitte, Kim Wuyts, Laurens Sion, Dimitri Van Landuyt, Ivo Emanuilov, Peggy Valcke, and Wouter Joosen. A comparison of system description models for data protection by design. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 1512–1515, 2019.

[108] Vanessa Ayala-Rivera and Liliana Pasquale. The Grace Period Has Ended: An Approach to Operationalize GDPR Requirements. In *2018 IEEE 26th International Requirements Engineering Conference (RE)*, pages 136–146, 2018.

[109] Johannes Heurix, Peter Zimmermann, Thomas Neubauer, and Stefan Fenz. A taxonomy for privacy enhancing technologies. *Computers & Security*, 53:1–17, 2015.

[110] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International journal of uncertainty, fuzziness and knowledge-based systems*, 10(05):557–570, 2002.

[111] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *Acm transactions on knowledge discovery from data (tkdd)*, 1(1):3–es, 2007.

[112] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd international conference on data engineering*, pages 106–115. IEEE, 2006.

[113] R.J. Anderson and F.A.P. Petitcolas. On the limits of steganography. *IEEE Journal on Selected Areas in Communications*, 16(4):474–481, 1998.

[114] David Chaum and Eugène Van Heyst. Group signatures. In *Advances in Cryptology—EUROCRYPT'91: Workshop on the Theory and Application of Cryptographic Techniques Brighton, UK, April 8–11, 1991 Proceedings 10*, pages 257–265. Springer, 1991.

[115] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings 23*, pages 506–522. Springer, 2004.

[116] Reza Shokri, George Theodorakopoulos, Carmela Troncoso, Jean-Pierre Hubaux, and Jean-Yves Le Boudec. Protecting location privacy: optimal strategy against localization attacks. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, page 617–627, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450316514.

[117] Jaap-Henk Hoepman. Privacy design strategies. In *IFIP International Information Security Conference*, pages 446–459. Springer, 2014.

[118] Michael Colesky, Jaap-Henk Hoepman, and Christiaan Hillen. A critical analysis of privacy design strategies. In *2016 IEEE security and privacy workshops (SPW)*, pages 33–40. IEEE, 2016.

[119] Jan Porekar, Aljosa Jerman-Blazic, and Tomaz Klobucar. Towards organizational privacy patterns. In *Second International Conference on the Digital Society*, pages 15–19. IEEE, 2008.

[120] Florian Kelbert and Alexander Pretschner. Towards a policy enforcement infrastructure for distributed usage control. In *Proceedings of the 17th ACM symposium on Access Control Models and Technologies*, pages 119–122, 2012.

[121] Siani Pearson and Yun Shen. Context-aware privacy design pattern selection. In *International Conference on Trust, Privacy and Security in Digital Business*, pages 69–80. Springer, 2010.

[122] Seda Gurses and Joris Van Hoboken. Privacy after the agile turn. *Cambridge Handbook of Consumer Privacy*, 2017.

[123] Frank Pallas, Katharina Koerner, Isabel Barberá, Jaap-Henk Hoepman, Meiko Jensen, Nandita Rao Narla, Nikita Samarin, Max-R. Ulbricht, Isabel Wagner, Kim Wuyts, and Christian Zimmermann. Privacy Engineering From Principles to Practice: A Roadmap. *IEEE Security & Privacy*, 22(2):86–92, 2024.

[124] Frederick Brooks and H Kugler. No silver bullet: Essence and accidents of software engineering. *IEEE Computer*, 20(4):10–19, 1987.

[125] Knut-H Rolland and Ole Hanseth. Managing path dependency in digital transformation processes: A longitudinal case study of an enterprise document management platform. *Procedia Computer Science*, 181:765–774, 2021.

[126] Claes Wohlin and Aybüke Aurum. Towards a decision-making structure for selecting a research design in empirical software engineering. *Empirical Software Engineering*, 20:1427–1455, 2015.

[127] Juhani Iivari. A paradigmatic analysis of information systems as a design science. *Scandinavian journal of information systems*, 19(2):5, 2007.

[128] I Niiniluoto. *Critical scientific realism*. Oxford University Press, 1999.

[129] Wanda J Orlikowski and C Suzanne Iacono. Research commentary: Desperately seeking the "IT" in IT research—A call to theorizing the IT artifact. *Information systems research*, 12(2): 121–134, 2001.

[130] Ilkka Niiniluoto. The aim and structure of applied research. *Erkenntnis*, 38(1):1–21, 1993.

[131] Noah Lemos. *Common sense: A contemporary defense*. Cambridge University Press, 2004.

[132] Alan R Hevner. A three cycle view of design science research. *Scandinavian journal of information systems*, 19(2):4, 2007.

[133] Vahid Garousi, Markus Borg, and Markku Oivo. Practical relevance of software engineering research: synthesizing the community's voice. *Empirical Software Engineering*, 25:1687–1754, 2020.

[134] Claes Wohlin, Aybuke Aurum, Lefteris Angelis, Laura Phillips, Yvonne Dittrich, Tony Gorschek, Hakan Grahn, Kennet Henningsson, Simon Kagstrom, Graham Low, Per Rovegard, Piotr Tomaszewski, Christine van Toorn, and Jeff Winter. The Success Factors Powering Industry-Academia Collaboration. *IEEE Software*, 29(2):67–73, 2012.

[135] Hugh Munby. Reflection-in-action and reflection-on-action. *Current issues in education*, 9(1): 31–42, 1989.

[136] Mike Hintze. Viewing the GDPR through a de-identification lens: a tool for compliance, clarification, and consistency. *International Data Privacy Law*, 8(1):86–101, 2018.

[137] Hugo Lefeuvre, Nathan Dautenhahn, David Chisnall, and Pierre Olivier. SoK: Software Compartmentalization. In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 3107–3126, 2025.

[138] Shukun Tokas, Olaf Owe, and Toktam Ramezanifarkhani. Language-based mechanisms for privacy-by-design. In *IFIP International Summer School on Privacy and Identity Management*, pages 142–158. Springer, 2019.

[139] Jukka Ruohonen. A Rapid Review Regarding the Concept of Legal Requirements in Requirements Engineering. *arXiv preprint arXiv:2509.06012*, 2025.

[140] Jukka Ruohonen, Kalle Hjerppe, and Eun-Young Kang. A Mapping Analysis of Requirements Between the CRA and the GDPR. In *2025 IEEE 33rd International Requirements Engineering Conference Workshops (REW)*, pages 215–222, 2025.

[141] Nguyen Hoang Thuan, Andreas Drechsler, and Pedro Antunes. Construction of design science research questions. *Communications of the Association for Information Systems*, 44(1):20, 2019.

[142] Bernhard Dachs, Iulia Siedschlag, Weijie Yan, Maria Yoveska, Fernanda Boeira, and Sean Ivory. Study to map, measure and portray the EU mid-cap landscape. *Publications Office of the European Union*, 2022.

[143] Elias Grünewald, Paul Wille, Frank Pallas, Maria C. Borges, and Max-R. Ulbricht. TIRA: An OpenAPI Extension and Toolbox for GDPR Transparency in RESTful Architectures. In *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 312–319, 2021.

[144] European Commission. A simpler and faster Europe: Communication on implementation and simplification, 2025.

[145] European Commission. Proposal for a REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL amending Regulations (EU) 2016/679, (EU) 2016/1036, (EU) 2016/1037, (EU) 2017/1129, (EU) 2023/1542 and (EU) 2024/573 as regards the extension of certain mitigating measures available for small and medium sized enterprises to small mid-cap enterprises and further simplification measures, 2025.

[146] Harshvardhan J Pandit. Simple Now, Complex Later: The Questionable Efficacy of Diluting GDPR Article 30 (5). In *Privacy Technologies and Policy: 13th Annual Privacy Forum, APF 2025, Frankfurt am Main, Germany, October 22–23, 2025, Proceedings*, page 127. Springer Nature, 2025.

TURUN
YLIOPISTO
UNIVERSITY
OF TURKU

Painosalama, Turku, Finland 2026

ISBN 978-952-02-0592-8 (PRINT)
ISBN 978-952-02-0593-5 (PDF)
ISSN 2736-9390 (PRINT)
ISSN 2736-9684 (ONLINE)