The Journal of Systems and Software 216 (2024) 112144

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

GDPR compliance via software evolution: Weaving security controls in software design☆

Vanessa Ayala-Rivera [a,b], A. Omar Portillo-Dominguez [a,*], Liliana Pasquale [c]

[a] Technological University Dublin (TU Dublin), Ireland
[b] National College of Ireland (NCI), Ireland
[c] University College Dublin (UCD) & the SFI Research Centre for Software (Lero), Ireland

ARTICLE INFO

Keywords:
Software evolution
Privacy and security
Software design
Data protection
GDPR
Compliance

ABSTRACT

Software should comply with international privacy laws, like the General Data Protection Regulation (GDPR). However, implementing appropriate technical controls is often an error-prone and time-consuming process. This is partly due to the limited knowledge of software engineers about privacy and security. This paper proposes SoCo, a semi-automated approach to support organizations in achieving software compliance with the GDPR data protection principles. To do so, SoCo supports engineers in identifying and integrating appropriate technical controls in sequence diagrams during the design phase. SoCo includes a technique to assist engineers to identify data processing activities in software applications modeled as sequence diagrams that may need to comply with the GDPR, a catalog of privacy and security controls that engineers can use to fix non-compliant activities, and a technique to implement such controls in the non-compliant sequence diagrams. Our evaluation results show that SoCo can help software engineers identify and design appropriate security controls to address GDPR violations and required moderate manual effort when applied to a substantive open-source application.

Editor's note: Open Science material was validated by the Journal of Systems and Software Open Science Board.

1. Introduction

Following the enforcement of the EU General Data Protection Regulation (GDPR) in 2018, new and revised data protection laws have come into effect in several countries worldwide (International Association of Privacy Professionals, 2024), such as the Brazilian General Data Protection Law (LGPD) (Brazilian Federal Government, 2018), the California Consumer Privacy Act (CCPA) (California State Legislature, 2018), the New Zealand Privacy Act (New Zealand Parliamentary Counsel Office, 2020), and China's Personal Information Protection Law (People's Republic of China, 2020). These laws seek to ensure that best practices are followed by organizations when processing personal data. Such a rapidly changing regulatory environment requires organizations to evolve their software systems to integrate technical controls that can fulfill the new obligations imposed by these privacy laws.

However, data breaches persist mainly due to the inadequate implementation or lack of robust technical controls in modern software systems (New Zealand Government, 2020). A personal data breach can occur when an organization suffers a security incident that leads to any accidental or unlawful destruction, loss, alteration, disclosure of, or access to personal data (DPC Ireland, 2019). This demonstrates that, although security and privacy are intertwined, they are both required to comply with data protection (Herzog, 2016).

Software engineers, who are often tasked to achieve compliance in software systems, may not have sufficient expertise to develop appropriate technical privacy and security controls (hereafter referred to as priv&sec controls) (Hadar et al., 2018; Dias Canedo et al., 2020). Existing studies have confirmed that software engineers (hereafter referred to as engineers) often experience difficulties in implementing security (Votipka et al., 2020; Rauf et al., 2022) and privacy requirements (Senarath and Arachchilage, 2018; Peixoto et al., 2020; Alhazmi and Arachchilage, 2021) in software systems. Thus, they can make errors while selecting and implementing priv&sec controls to fulfill data protection obligations (Votipka et al., 2020). In large and complex systems, it is even more challenging to identify where and how priv&sec controls should be implemented. For example, Facebook, GitHub, and Twitter have faced security issues in their systems, leading to the exposure of passwords in plain text (Krebs, 2019).

☆ Editor: Patricia Lago.

* Corresponding author.
E-mail addresses: vanessa.ayalarivera@tudublin.ie (V. Ayala-Rivera), omar.portillo@tudublin.ie (A.O. Portillo-Dominguez), liliana.pasquale@ucd.ie (L. Pasquale).

https://doi.org/10.1016/j.jss.2024.112144
Received 14 June 2023; Received in revised form 22 March 2024; Accepted 18 June 2024
Available online 21 June 2024

Although roadmaps (Mcadam, 2017; Deloitte, 2017), checklists (DPC Ireland, 2017; Isaca, 2017), and processes (Coleti et al., 2020; Guamán et al., 2021) have been proposed to support GDPR compliance, they need to provide explicit guidance about which priv&sec controls to implement and how. These approaches are also typically detached from software engineering practices; thus, they can hardly be adopted during the software development life cycle (SDLC) (Martin and Kung, 2018; Ehécatl Morales-Trujillo et al., 2019). To tackle these problems, this paper proposes SoCo, a semi-automated approach to support engineers, during the design phase of a SDLC, in integrating priv&sec controls within sequence diagrams of software design for processing personal data securely and ensuring GDPR compliance. Design artifacts allow engineers analyzing a system from various perspectives (e.g., architectural, functional); while, from a security and privacy point of view, sequence diagrams can help engineers to follow the flow of data in a system to identify risks arising from the processing of personal data.

SoCo provides a technique to create data flow filters which can be used by data protection (DP) experts and engineers to selectively retrieve a subset of sequence diagrams of interest (e.g., those that refer to specific data processing activities or personal data types) to assess compliance. Once non-compliant data flows have been identified, SoCo assists engineers in the selection of appropriate priv&sec controls to fix the GDPR violations found in those sequence diagrams. To achieve this aim, we provide engineers with a structured catalog of priv&sec controls, where each control is associated with the GDPR data protection principles (DPPs) that it supports and a possible way to implement the control. SoCo also provides engineers with an automated technique to evolve non-compliant diagrams by injecting an implementation of the selected priv&sec controls. We reuse concepts from Aspect Oriented Programming (AOP) to specify the conditions under which a priv&sec control should be injected and the implementation of priv&sec controls. The evolved diagrams explicitly guide engineers regarding what controls should be implemented and where. Finally, we provide a prototype tool that implements the techniques and artifacts proposed, along with the material used in our evaluation (Ayala-Rivera et al., 2023). Together, the tool and approach help advance the knowledge in the discipline of secure software engineering and privacy-by-design. They accomplish this by providing practical artifacts and novel techniques that engineers can use to select priv&sec effectively controls to ensure GDPR compliance and efficiently apply them to design artifacts of software systems. Ultimately, the tool and approach facilitate the evolution of such systems to achieve GDPR compliance.

Our work focuses on the GDPR (European Parliament, Council of the European Union, 2016), as it is usually viewed as the "*gold standard for data protection*". In particular, we focus on Article 5, which lists seven DPPs that must be considered when processing personal data (shown in Table 1). Data protection laws, similar to the GDPR, have been enacted in various regions (e.g., CCPA in California, and LGPD in Brazil). As the GDPR DPPs are shared across many of these laws, we expect SoCo to be applicable beyond the GDPR. Although SoCo supports engineers in correcting GDPR violations, it does not provide formal guarantees that software systems evolved using it will be compliant with GDPR.

To evaluate our work, we conducted a case study to assess the manual effort necessary to apply SoCo to a large-scale software system. We asked an IT professional to apply SoCo to FenixEdu (Instituto Superior Tecnico, 2019), a substantive software used in higher education institutions. SoCo allowed the participant to apply appropriate priv&sec controls on 285 data flows that were considered relevant to the GDPR. SoCo also required moderate manual effort when applied to a large-scale software system. Our results also indicate that SoCo can yield significant time savings for the engineers conducting GDRP compliance tasks.

The rest of the paper is organized as follows: Section 2 presents the background and related work. Section 3 motivates our work using a running example. Section 4 describes our approach. Section 5 describes our prototype. Section 6 presents our evaluation and its results. Section 7 presents a final discussion of our approach and ideas for future work. Section 8 discusses the threats to validity, and Section 9 concludes the paper.

2. Background and related work

In this Section, we discuss two related areas to our work: GDPR compliance and privacy-by-design. We also discuss previous work that, similarly to our approach, uses AOP to implement priv&sec controls.

2.1. GDPR compliance

Achieving compliance with any data protection law is an organizational-wide effort. This task typically involves different stakeholders, such as data users, DP experts, and engineers, when compliance relates to implementing changes to software systems. Regardless of the organization, one of the first steps towards GDPR compliance is conducting a data mapping exercise to create a *data inventory* (Martin and Kung, 2018; Piras et al., 2021). This exercise aims to identify what personal data elements are processed and how personal data flows in the organization. Using this information, DP experts can review the data processing activities to identify those that violate the GDPR. Once the non-compliant activities have been identified, engineers are responsible for implementing appropriate priv&sec controls to fix the violations. Achieving compliance is an ongoing process, as data protection laws and software systems usually evolve.

Several commercial solutions have been developed to help organizations achieve compliance with the GDPR, such as proprietary roadmaps (Mcadam, 2017; Deloitte, 2017; SAS, 2018), off-the-shelf privacy technology tools (IAPP, 2022), and self-assessment checklists to evaluate the level of GDPR-readiness of an organization (DPC Ireland, 2017; Isaca, 2017). However, these solutions are proprietary and cannot be afforded by small organizations. They are also detached from software engineering practices (Martin and Kung, 2018).

The software engineering research community has explored approaches focused on elicitation and analysis of requirements for GDPR compliance (Aberkane et al., 2021; Alkubaisy et al., 2021), assessment and demonstration of compliance (Torre et al., 2020; Cortina et al., 2021), and achievement and maintenance of compliance (Kabanov, 2016; Rhahla et al., 2021). Additionally, some work focuses on particular aspects of compliance, such as privacy policy completeness (Cejas et al., 2021), consent verification (Robol et al., 2022), compliance with personal data management (Truong et al., 2019) and data processing agreement requirements (Amaral et al., 2021), as well as formal modeling of processing purposes (Vanezi et al., 2020). Despite the growing literature on compliance, existing solutions do not focus on how to automatically select priv&sec controls to ensure GDPR compliance and apply these controls to the design artifacts of software systems.

2.2. Privacy-by-design

Many general-purpose principles and frameworks have been proposed to help integrate privacy capabilities into systems. Ann Cavoukian (former Information & Privacy Commissioner of Ontario, Canada) proposed the seven foundational privacy-by-design principles (Cavoukian et al., 2009). While these principles embody the modern conceptualization of data protection through technology design, the need for more specificity has hampered their adoption by engineers (Bu et al., 2020). Other international organizations have also developed privacy guidelines, such as the five privacy practice principles (FIPPs) (Federal Trade Commission, 1996) by the Federal Trade Commission and the OECD privacy framework (OECD, 2022), which have served as a reference point to governments. Privacy-by-design strategies have also been proposed to offer more concrete solutions that engineers can implement. Hoepman (2014) presents eight privacy design strategies to support IT architects considering privacy-by-design in the SDLC. Although these approaches offer a comprehensive set of best practices to consider privacy during system design, they do not suggest to engineers how to implement and integrate priv&sec controls in software systems. A recent literature review (Andrade et al., 2023)

Table 1
GDPR data protection principles.

#	DPP	Description
1	Lawfulness, Fairness, and Transparency	Data processing activities must have a valid legal basis, must not unfairly disadvantage individuals, and must be conducted transparently.
2	Purpose limitation	Personal data should be collected for specified and explicit purposes and should not be used in a manner incompatible with such purposes.
3	Data minimization	Personal data should be adequate, relevant, and limited to what is necessary for the purposes for which it is processed.
4	Accuracy	Personal data should be accurate and, where necessary, kept up to date.
5	Storage limitation	Personal data should be kept in a form that permits identification of data subjects for no longer than is necessary for the purposes of its processing.
6	Integrity and Confidentiality	Personal data should be processed in a manner that ensures appropriate security (e.g., protection against unauthorized processing and accidental loss).
7	Accountability	Data controllers are responsible for GDPR compliance and must be able to prove it (e.g, maintaining records of processing activities).

also flags the limited number of models, processes, and tools to support Privacy by Design throughout the SDLC.

Finally, LINDDUN (Deng et al., 2011; Wuyts and Joosen, 2015) is a privacy threat modeling framework that contributes to achieving some privacy-by-design principles. It supports systematic, proactive elicitation and mitigation of privacy threats ("Proactive, not Reactive" principle) from the system architecture represented using data-flow diagrams ("Privacy Embedded into the design" principle). The assessment will pinpoint any design and implementation issues that require countermeasures. LINDDUN attempts to achieve the "Privacy By Default" principle by including non-compliance privacy threats that refer to violations against the GDPR data protection principles. LINDDUN also achieves the "Visibility and Transparency" principle by documenting the identified privacy threats, ensuing countermeasures, and the threat analysis process itself. Our approach aims to achieve the "Embedded into the design" principle by providing practical artifacts and novel techniques that engineers can use to select priv&sec controls to ensure GDPR compliance and efficiently apply them to sequence diagrams of software systems. We also address the "Privacy By Default" principle by focusing on compliance with the GDPR data protection principles. Since our approach focuses on software evolution to achieve GDPR compliance, we do not address the "Practice, not Reactive" principle. Also, our approach does not focus on the "Visibility and Transparency" principle. Unlike our work, although LINDDUN offers a taxonomy of mitigation strategies for privacy threats, it does not provide an automated approach to select and integrate mitigation strategies in the software system architecture. Finally, since LINDDUN operates on data-flow diagrams, it does not allow considering operation sequencing, which is relevant when incorporating priv&sec controls in the software design.

2.3. Aspect-oriented approaches for secure software

Previous research has suggested using AOP to integrate security aspects in software at various levels during software design (Mouheb et al., 2016) and implementation (Ali et al., 2015; Bodorik et al., 2014). Similarly to these approaches, we use AOP to inject a specification of priv&sec controls in the design artifacts. Unlike previous work, SoCo addresses GDPR regulatory compliance issues by automatically injecting priv&sec controls in the sequence diagram representing the data processing activities violating the GDPR. Also, SoCo provides a more intuitive textual notation for engineers to configure the controls and does not require specialized knowledge such as meta-modeling experience. Finally, SoCo provide a GUI to visualize the evolved diagrams.

3. GDPR compliance in a Student Information System

Our example involves a Student Information System (SIS) used in a university (UniXYZ) to process student data for academic purposes. A possible data flow is shown in the diagram in Fig. 1. An academic staff requests the SIS to change a student's grade. Next, the SIS verifies the credentials of the academic staff in the database (DB). If authorized, the SIS modifies the student's grade and confirms the change. Otherwise, it notifies the academic staff that the request was denied. A problematic situation might arise if a module coordinator opens a dispute about an incorrect grade (because it is different from the grade they originally proposed), but the SIS does not log enough data to prove why and who changed the grade to support a grade rectification. This violates the Accuracy DPP, which claims that "*every reasonable step must be taken to ensure that personal data that are inaccurate ... are erased or rectified without delay*". A possible priv&sec control to fix this issue is to log any processing activity performed to a student record, to support postmortem event reconstruction and justify grade rectification.

The following is another example of a possible GDPR violation in UniXYZ's SIS. Let us suppose the SIS fails to provide students with a clear consent mechanism and instead assumes their implicit consent for reusing their information for non-directly teaching-related activities like automatically being added to marketing communications when registering to a course (as illustrated in Fig. 2). This scenario would involve two different GDPR non-compliances: On the one hand, personal information is reused for marketing without notifying individuals or obtaining their explicit consent to use it in this way. This violates the Lawfulness, Fairness, and Transparency DPP as organizations must be transparent about how the collected data will be used and also give the individuals the option to provide informed consent using affirmative action. The scenario also violates the Purpose Limitation DPP, as such information reuse is incompatible with the consent initially provided by the individuals to process their information for teaching-related activities.

It is important to highlight that identifying what controls should be implemented to achieve GDPR compliance is difficult. DPPs are expressed using vague and abstract terminology (e.g., "*every reasonable step*", "*without delay*"), so they can have multiple interpretations and provide limited information about what and how priv&sec controls should be enforced. To identify possible GDPR violations, engineers typically must manually identify and inspect the sequence diagrams where personal data processing occurs. This is cumbersome and error-prone, especially in large systems where there may be multiple diagrams to be examined. Finally, engineers may not have the expertise to

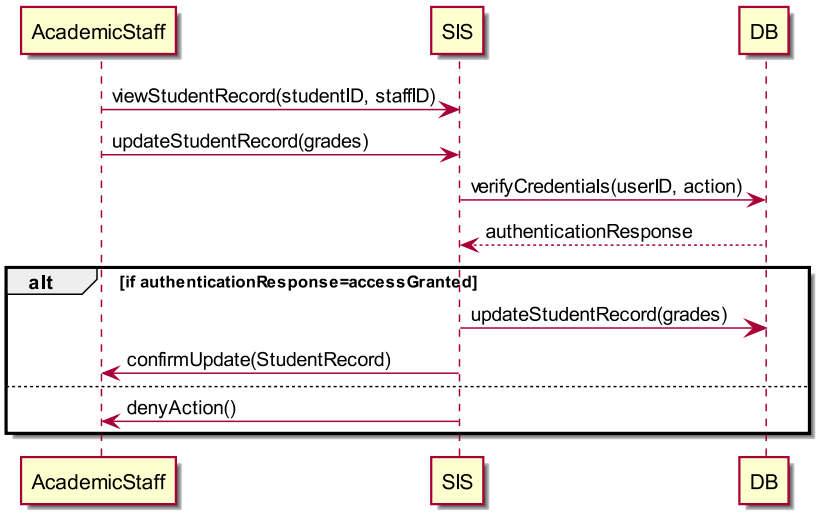


Fig. 1. Grade change diagram.

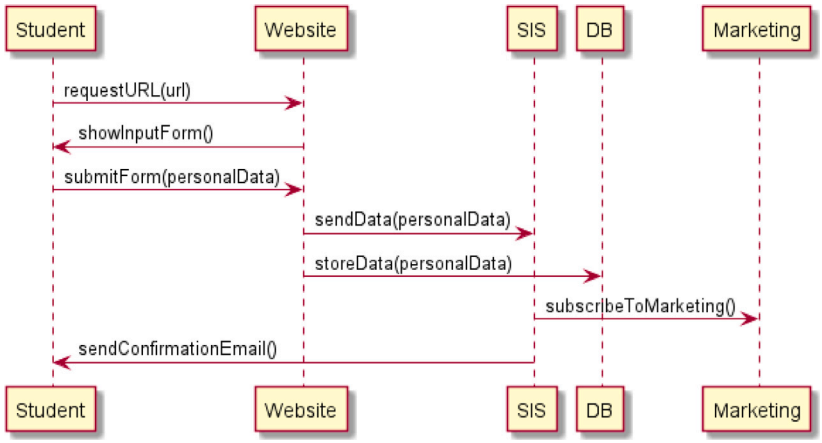


Fig. 2. Student curse registration diagram.

determine and apply appropriate priv&sec controls (Martin and Kung, 2018; Lopez et al., 2019; Alhazmi and Arachchilage, 2021), which can avoid GDPR violations.

4. SoCo - A software evolution approach for compliance

We propose a semi-automated approach (SoCo) to support organizations in achieving compliance with their software systems with the GDPR DPPs. Fig. 3 shows the steps of our approach and their inputs and outputs. SoCo supports engineers by selecting which priv&sec controls, and where, should be implemented to address the GDPR DPPs violations. The selected priv&sec controls are injected automatically within the sequence diagrams of software design representing data flows violating the GDPR.

4.1. SoCo's inputs

As shown in Fig. 3, SoCo requires two inputs: a data dictionary and the sequence diagrams suspected of violating the GDPR.

4.1.1. Data dictionary

As part of a GDPR compliance exercise, DP experts need to gather information about all the data assets held by the organization and the processing activities within the organization, which are typically documented in a *data inventory* (Piras et al., 2021). For SoCo, we require the data inventory to be represented as a list of pieces of personal data (elements) included in information systems within the organization (hereafter referred to as a data dictionary). It is worth pointing out that conducting an organization-wide data audit is a fundamental step to comply with the GDPR, so it is normally part of any compliance exercise (Piras et al., 2019) (and good practice of data governance). Thus, we consider it reasonable to assume a data dictionary will already exist in the organization to be leveraged by SoCo.

An example is shown in Fig. 4. A data dictionary comprises: (1) Keywords, which are the words that serve as the standard terminology used in SoCo. They identify the data items or operations derived from the data inventory or the software system. (2) Related terms, which are words that have some connection among them and with their associated keyword. They can be synonyms of a keyword or words grouped intentionally. This <keyword-related term> format allows multiple terms to be grouped under the same key. This format allows identifying and fixing GDPR violations uniformly for similar diagrams while avoiding term inconsistencies. For instance, in the case of synonyms, they are used for supporting search and pattern matching in SoCo. For example, suppose an engineer defines the term *erase* as an operation of interest in a diagram. In that case, SoCo will also be able to detect automatically other similar operations such as *cancel* or *delete* (assuming they have been defined as related terms) and consider them during the filtering of the diagrams. This prevents diagrams applying similar data processing operations from being handled differently. The same applies to words grouped intentionally, such as special category data like "facial image" and "retinal scan". Grouping them allows

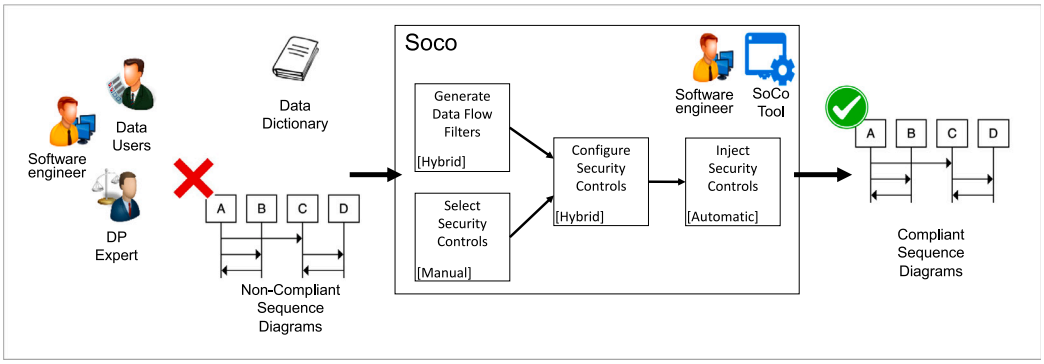


Fig. 3. Overview of SoCo.

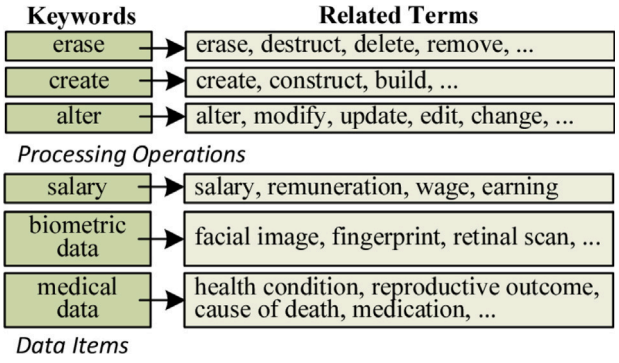


Fig. 4. An example of a data dictionary.

using the same priv&sec controls in all diagrams processing that data type. The data dictionary is represented as a .CSV file where each row represents a key (the first comma-separated value) and its related terms (from the second term onwards).

4.1.2. Non-compliant diagrams

As part of a GDPR-compliance exercise, engineers and DP experts need to identify which data processing activities documented in the data inventory violate the GDPR DPPs. Since we focus on software systems, we require the UML sequence diagrams of the non-compliant processing activities as input to SoCo.

We selected UML sequence diagrams as our notation to model scenarios for the following reasons. UML is the de facto standard for modeling and design of software systems in terms of structure and behavior (James, 2013). In particular, UML sequence diagrams have been widely adopted by both practitioners and researchers in the software engineering field as a standard language for object-oriented analysis and design. They are easy to understand, which facilitates communication between business units and engineers. They also clearly show not only the data flow within a system, but also the dynamic behavior and interaction between components in the system. Thus, stakeholders can better visualize how users, or external entities, interact with the system and how the system responds. Likewise, UML sequence diagrams provide the level of detail needed to understand the precise sequence of events and how objects interact during a specific data processing activity. All these elements facilitate that in our case, engineers and DP experts can use them to identify risks arising out of the processing of personal data.

We also considered other alternatives to UML sequence diagrams such as Data Flow Diagrams (DFD), flowcharts, and state diagrams. While DFDs also offer a visual representation of information flows in a system, these flows are modeled from an architectural perspective. Thus, they do not indicate the sequence of data movement. Meanwhile, flowcharts illustrate the sequence of steps or decision points in a process. However, they do not model how objects interact with each other. Finally, state diagrams are used to model the dynamic behavior of a complex system focusing on the states and transitions of objects. So, they are limited to illustrating the interactions and message flows between objects or components in a system.

We assume that sequence diagrams already exist as part of the documentation of the software system (i.e., either they were created as part of the design phase – as dictated by good software engineering practices –, or were reverse-engineered from the source code of the software system – as there are plenty of tools available that can help generate this type of diagrams automatically (Visual Paradigm, 2023; Altova GmbH, 2023) –). Finally, we represent the sequence diagrams using PlantUML, as it uses a simple textual notation to specify UML diagrams (PlantUML, 2023).

SoCo requires the non-compliant diagrams as an input because identifying GDPR violations and properly interpreting the law requires subject–matter expertise from a legal team (DP experts). Also, assessing GDPR compliance of data processing activities is required irrespective of the strategy used to identify GDPR violations (e.g., by analyzing existing use cases, or generating abuse and misuse cases through privacy or security threat modeling methodologies like LINDDUN (Deng et al., 2011) or STRIDE (Hernan et al., 2006)). Thus, we assume that this exercise has been conducted and therefore, the non-compliant activities have been identified, and their corresponding sequence diagrams are available. Nevertheless, the selection of non-compliant sequence diagrams given to SoCo as input does not need to be completely accurate. That is, it is sufficient to input any diagrams that might violate a GDPR DPP. This is because SoCo can help identify diagrams that are out-of-scope of the GDPR, as they will not contain relevant personal data items (i.e., those documented in the data dictionary) and can be simply discarded using the data flow filters during SoCo's first step. For instance, engineers can create a single data flow filter composed of all the elements in the data dictionary, and when this filter is applied over the diagrams, any diagram which was not selected, can be safely classified as GDPR irrelevant. Similarly, even though GDPR-compliant diagrams are input to SoCo perhaps by mistake or suspicion of non-compliance, these will not be modified/evolved by SoCo because they will not qualify for any of the configured injection rules.

4.2. SoCo's steps

As shown in Fig. 3, SoCo performs four main steps. It generates a data flow filter and selects, configures, and injects priv&sec controls. Although all steps are supported by the prototype tool, different degrees of manual human interaction are required. This has been represented in the figure by tagging the steps with the labels *manual* (if the step is mainly done by the user by utilizing the tool), *hybrid* (if the step is driven by human inputs, but most work is done by the tool), and *automatic* (if the step is done by the tool, and the user only needs to trigger it).

4.2.1. Generation of data flow filters

To facilitate the assessment of the diagrams for GDPR compliance, SoCo can generate data flow filters that the engineers and DP experts can utilize to filter the diagrams into smaller subsets that are more manageable to inspect, especially in large software systems. The filters provide a convenient way of retrieving the diagrams of interest for analysis, as they can refer to specific data items, processing activities, or even system components or actors. To achieve this aim, we provide a technique which converts the keys and the related terms of the data dictionary into searchable elements.

We create a four-level taxonomy composed of all the unique terms in the data dictionary. The first level is the root and groups all the possible data filter elements (e.g., terms and attribute types). The second level encompasses all attribute types (e.g., personal data items, data processing operation, processing purpose). The third level comprises the keywords specified in the data inventory, while the fourth level comprises their associated related terms.

After the taxonomy is created, engineers can generate a data flow filter by selecting the various elements available at different granularity levels in the taxonomy. Several filters can be made for analyzing the processing activities in the various departments (e.g., those corresponding to marketing) or software components (e.g., those involving authentication). Thus, our tool provides a way to name and store the filters for later retrieval. When a filter is applied, the values of the selected taxonomy elements are matched against the names of classes, methods, and parameters in the sequence diagrams. To do this, we use a syntactic matching approach. We use "contains" (ignoring case) as the primary function to search for the filter values in the diagrams. However, other functions can be used, such as distance measures (Cohen et al., 2003) or semantic similarity (Jatnika et al., 2019). We assume that this matching approach is sufficient for finding relevant sequence diagrams since engineers usually follow meaningful programming naming conventions (e.g., verbs, nouns) (Butler et al., 2015). A data filter selects a diagram if at least one message exchange in the diagram contains each of the data filter's terms chosen as per the following rules: (1) If a selected taxonomy node has children (i.e., it belongs to any of the top three levels), all its children are considered part of the data filter. (2) Node values that share the same parent are converted into logical OR conditions for filtering purposes. That is, it is enough that one of the terms is present in a diagram for this to be returned as part of the data filter's results. (3) Nodes that do not share the same parent (e.g., a set of related terms belonging to different keywords) are converted into logical AND conditions. All the terms must be present in a diagram for this to be returned as part of the data filter's results. For instance, the diagram shown in Fig. 1 would be returned as a result of a data filter composed of one processing operation (i.e., update) and two personal data items (i.e., student record and grade).

An example of a data flow filter is shown in Fig. 5. The data flow filter generated (boxed in red) can retrieve all the processing activities that involve altering a student record to conduct course assessments. This criterion matches the scenario described in Section 3. Filters can be refined with more granular data items or mapped to synonyms to cover various analysis tasks. For example, in Fig. 5, *student record*s can be divided into individual data items; and the *alter* operation can be complemented with *modify* or *update* operations.

In addition to the terms available in the data dictionary, engineers can also consider in a data filter technical personal data items (e.g., cookie, IP address) or operations only present in a software system (e.g., authenticate), which might not have been documented during the generation of the data inventory/dictionary. These terms can be manually added as *tokens* to SoCo to be considered as searchable elements. These tokens are grouped in a separate category in the taxonomy (i.e., *Token* type) as part of the second level and follow the same search rules explained previously. In SoCo, the personal data items, processing operations, processing purposes, and/or token values that make a diagram qualify as data-filter-relevant are highlighted using the green color within the diagram to facilitate their visualization and analysis.

4.2.2. Selection of Priv&Sec controls

In this paper, we propose using a catalog of security and privacy controls that can serve as potential technical solutions to satisfy GDPR requirements in software systems. While privacy controls ensure that personal data is handled properly (i.e., not misused), security controls ensure that data is protected against unauthorized access/-modification. We initially gathered the controls through an extensive literature review of information technology standards for privacy/security compliance (e.g., ISOs) (ISO/IEC, 2024, 2022a), privacy-enhancing technologies (CIS, 2023; Cavoukian, 2012), and privacy design patterns (PRIPARE, 2016; Köffel et al., 2010; Colesky et al., 2016; Hoepman, 2014) as part of our previous work (Ayala-Rivera and Pasquale, 2018). This review was motivated by the need for more usable security tools and artifacts by engineers (Acar et al., 2016). The catalog currently has 47 controls. Each control contains a description of the control, the problem it can address, its type (i.e., security, privacy, or both), the gains obtained by implementing it, and the GDPR DPPs it best contributes to satisfying. Five experts in requirements, security and privacy engineering, and human–computer interaction validated the mapping between the GDPR DPPs and the controls (Ayala-Rivera and Pasquale, 2018).

To do this, the catalog was shared with each researcher (participant) who individually determined, based on their expert judgment, which controls were suitable to address each GDPR DPP. This was determined as a binary decision (i.e., considering the capability of the control to address a GDPR DPP and not by measuring the control's degree of appropriateness in satisfying the DPPs). The results from participants were consolidated, and a control was considered suitable to satisfy a DPP if most participants indicated it. It is worth noting that all participants were familiar with the GDPR terminology and its DPPs (as it was a criterion for their participation in the study). Also, they reported feeling confident about their decisions and spent a reasonable time performing the validation (ranging between 45 and 105 mins) as discussed in Ayala-Rivera and Pasquale (2018).

To develop SoCo, we focused exclusively on the ten controls shown in Tables 2 and 3. This is because they are some of the most used priv&sec controls covered by our catalog and a representative subset covering possible technical solutions to address all the GDPR DPPs. Also, we have purposely focused on security controls for the first version of SoCo because it is typically accepted that organizations cannot have effective privacy without achieving a basic foundation of information security first (NIST, 2020). In this work, we have extended the definition of each control by adding an implementation hint, which lists the software architecture components or business flows where the control may be suitable. For instance, the controls can suggest adding functionality to the system (e.g., authentication based on Single Sign-On -SSO-) and new components (e.g., SSO server). They can also modify functionality (e.g., replacing a message exchanged between components by encrypting the data transmitted). Controls can also recommend removing functionality (e.g., a biometric-based access control without having an appropriate legal basis). Using this catalog and the data flow filters, engineers can identify priv&sec controls to fix the GDPR DPP violations existing within the non-compliant diagrams.

Let us consider the following non-compliant scenario in UniXYZ: *Employees access multiple systems through the internal portal, logging in and out each time to switch systems. The problem is that some employees might reuse passwords which may not be strong. This is a risk to DPP #6 (Integrity and Confidentiality), as attackers might steal credentials and gain access to the organization with severe consequences.* To select a suitable control, an engineer can take the catalog, filter it by DPP #6, and look at the *implementation hints* to guide their decision. For example, priv&sec controls #40 (SSO) and #47 (Web Filtering) can address a violation of DPP #6. Then, the engineer determines that SSO is more appropriate as the implementation hint of this control better matches the scenario, that is, it involves login, access to an application, credentials, and authentication.

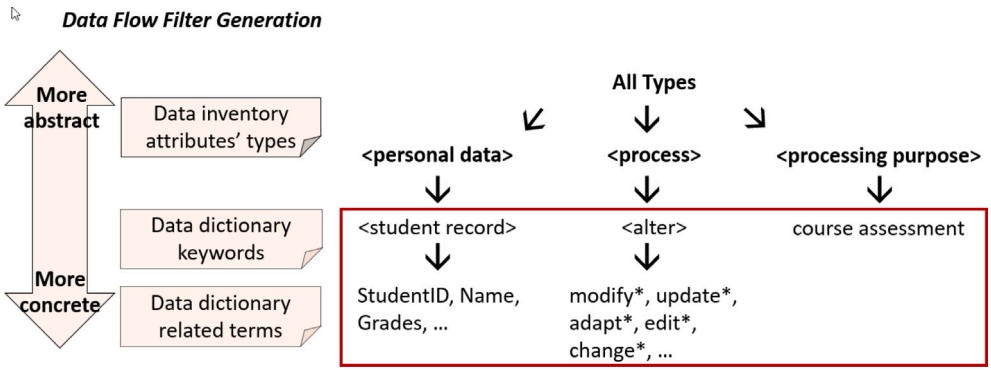


Fig. 5. Data flow filter for retrieving diagram in Fig. 1.

Table 2
Sample of SoCo priv&sec controls catalog (Part 1).

ID	Privacy and security controls	GDPR DPPs	Implementation hint	Type
6	**Attribute-Based Access Control (ABAC).** Access rights are granted through policies composed of attributes working together. Attributes, such as user and resource attributes, determine access. **Problem Addressed:** Prevent leak of information by revealing more information than needed. **Benefit:** Grants access based on the attributes of a software system and its users.	2, 3, 6	authorization, access, policies, attributes	S
9	**Cookies Consent Banner (GDPR Compliant).** A consent banner is a cookie notice, that appears on websites upon the user's first visit to the site. To give the users the possibility to opt in and opt out of the various types of cookies and then get their consent to the setting of cookies. **Problem Addressed:** Implied consent. **Benefit:** Users can make informed decisions about their privacy.	1, 2, 3	view, browser, website, cookies	P
13	**Database Auditing.** It involves tracking and logging events that occur on a database to be aware of the actions performed by the database users. Database administrators and consultants often set up auditing for security purposes, for example, to ensure that those without permission to access information do not access it. Moreover, organizations can demonstrate compliance with information security legislation and prevent fraud and other incidents. **Problem Addressed:** Demonstrate compliance, identify best practices, and discover risks. **Benefit:** Compliance improvement.	7	tracking, events, logging, database, auditing, access, compliance	S, P
17	**Encryption.** When data should be hidden from plain view, provide encryption mechanisms to scramble the contents of a message/file so that authorized users can only read it by unscrambling it. **Problem Addressed:** Unauthorized viewing. **Benefit:** Sensitive data, purchase, payment, access, store, view.	6	tracking, events, logging, database, auditing, access, compliance	S
20	**Logging.** The data controller implements logging to demonstrate compliance. **Problem Addressed:** Prevent non-compliant behavior. **Benefit:** The organization can demonstrate compliance with information security legislation and prevent fraud or other incidents.	4, 7	DB, privileges, system/access events, data changes, failures to access a resource, sensitive operations	P, S

4.2.3. Configuration of Priv&Sec controls

SoCo also helps engineers to configure the selected priv&sec controls (needed to address the GDPR DPP non-compliances), tailoring them to the target software system.

To specify the conditions under which priv&sec controls should be injected into the non-compliant diagrams, we provide the *implementation pattern template* shown in Fig. 6(a). An example of the template applied to a Logging control is shown in Fig. 6(b). The template uses placeholders that engineers can customize based on the priv&sec control and the system's architecture. It includes some *general information*, such as a name, a brief description of the control, and the GDPR DPPs the control best contributes to satisfy. Borrowing concepts from Aspect-Oriented Modeling, each control is also characterized by one or more pairs of *pointcut*(s) and *advice*.

The pointcut identifies where a priv&sec control should be injected in a non-compliant diagram. To achieve this aim, the engineer should specify the messages that need to occur in a sequence diagram to trigger the injection of priv&sec controls. A pointcut is specified by indicating the characteristics of the messages such as the action occurring (*ProcessingOperation*), the type of personal data exchanged (*DataType*), the total number of messages that need to be intercepted to recognize a scenario of interest (*NumOfMessages*), and the actors involved (*Actors*). For example, an engineer may determine that to support appealing of exam results, Logging should be added to track all the changes a user makes to student records in the database. Thus, a possible Logging control implementation may require intercepting one message involving a data alteration operation performed between an actor acting as a sender and a database acting as a receiver (as shown in Fig. 6(b)). Another implementation can intercept two messages, one involving a request to modify the data and another indicating denial of such request. This means that engineers are interested in only logging requests that modify data in the database that were denied.

The advice specifies the behavior that will be added to the diagram (i.e., the instructions that compose the implementation of a priv&sec control). An advice is characterized by a type (*AdviceType*) indicating (a) if the advice adds, removes, or modifies functionalities and (b) where the priv&sec actions will be inserted w.r.t. to the pointcut. The supported types of advice are shown in Fig. 7. *ADD_BEFORE*, *ADD_AFTER*, and *ADD_BETWEEN* add functionality to the system before, after, or between the intercepted messages. *WRAP_AROUND* adds

Table 3
Sample of SoCo priv&sec controls catalog (Part 2).

ID	Privacy/Security controls	GDPR DPPs	Implementation hint	Type
36	**Retention Policy.** It is an organizational established protocol to retain information for operational or regulatory compliance. The retention policy identifies the time the data should be kept or maintained, irrespective of format (paper, electronic, etc.). **Problem Addressed:** Indefinite storage of information. **Benefit:** Reduce legal risks, discovery costs, and recovery time associated with lawsuits.	5	store, database, policy	P
37	**Role-Based Access Control (RBAC).** It is a method for coarse-grained access control that is based on a person's role within an organization. It is possible to designate whether a person is a user, an administrator, or an end-user and align roles and access permissions with the person's positions in an organization. **Problem Addressed:** Leak of information by revealing more information than needed. **Benefit:** Permissions are granted only if a person them to perform their tasks.	2, 3, 6	authorization, access, role, permissions, privileges	S
40	**Single Sign-On (SSO).** It is an authentication process that allows a user to access multiple applications with one set of credentials. It eliminates further prompts when the user switches applications during the same session. SSO is commonly used in enterprises when clients access multiple resources connected to a local area network. **Problem Addressed:** Users need to sign in for every application they use. **Benefit:** It allows users to log in to different apps and resources using a single set of credentials.	6, 7	authentication, login, credentials, access application	S
46	**Valid Informed Consent.** Whenever data collection/disclosure needs to be legitimized, provide click-through agreements ("click and accept") to confirm the user's understanding or consent on an "as-needed basis" by using drag-and-drop functions for consenting data disclosure. **Problem Addressed:** Prevent users from accepting terms and conditions of service too quickly (due to long legal terms) without having read or understood what they consented to. **Benefit:** Users can make informed decisions about their privacy.	1, 7	View Component, advertising, marketing, registration, sharing	P
47	**Web Filtering.** A web filter is a program that can screen an incoming web page to determine if some or all of it should be displayed to the user. The filter checks the origin or content of the web page against a set of rules provided by the organization or the person who installed the web filter. **Problem Addressed:** Indefinite storage of information. **Benefit:** Prevents users from accessing websites considered inappropriate or harmful and protects an organization's assets from cyberattacks.	6, 7	internet, browser, standalone program, router, network, content-control	S

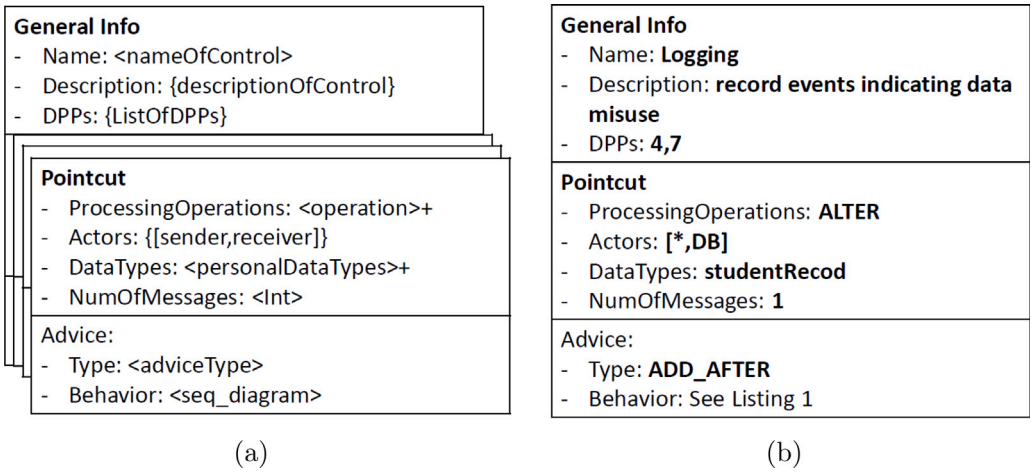


Fig. 6. (a) Template for defining priv&sec controls, and (b) Example of the template instanced for a Logging control.

functionality around the messages (e.g., adding conditional statements to process personal data only if consent was obtained). *REPLACE_BETWEEN* replaces functionality comprised between two message sequences. *REPLACE_ALL* replaces a message sequence with another (e.g., replacing a user-based access control with a role-based one). The advice behavior is specified using PlantUML instructions, which use a simple textual notation for UML diagrams (PlantUML, 2023).

The advice behavior of the Logging control example is presented in Listing 1. It forces a service in the system to ask the Logger to record a transaction. To do so, the Logger de-identifies the logging information and subsequently writes the transaction. As shown in Listing 1, it is possible to declare tokens for certain model entities (represented by words starting with $), so they can be replaced by specific component instances of the target system to facilitate the customization of the controls. For example, *"$service"* can refer to the SIS, whereas *"$transaction"* can refer to the object holding the details that will be used to modify a record in the database.

Listing 1: Behavior specification of a Logging control

```
$service -> Logger: writeLog($transaction)
Logger-> Logger: deidentifyLog($transaction details)
Logger-> Logger: writeTransactionLog(deIdentified $transaction)
```

The template provided for a priv&sec control can include more than one pointcut–advice pairs. These are used when implementing a control requires integrating behavior in different parts of a diagram. For example, an Encryption control implementation could involve two pointcut–advice pairs: one to detect when sensitive data is transmitted

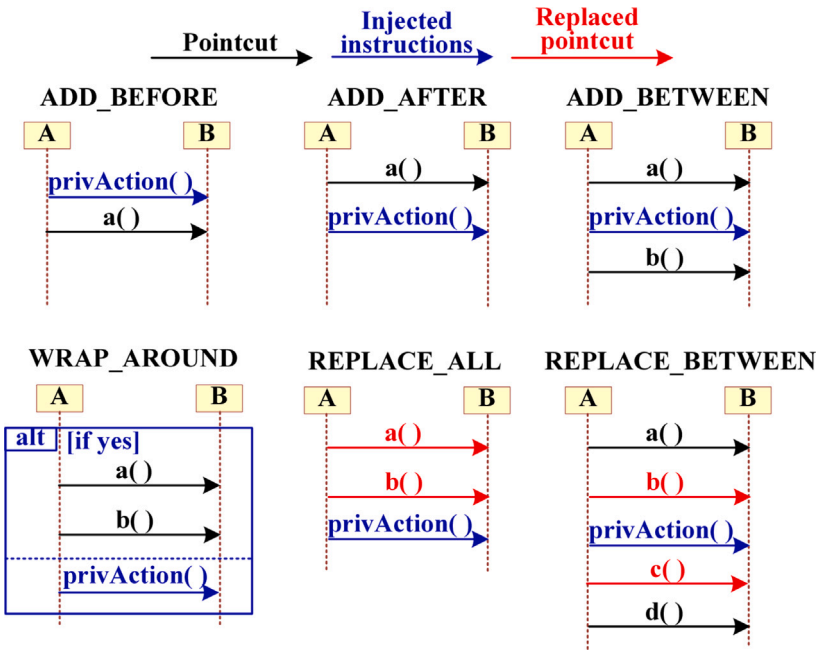


Fig. 7. Types of advice.

so that it can be encrypted and another to detect when data is used so that it can be decrypted.

We provide pre-filled templates for the ten priv&sec controls shown in Tables 2 and 3 (available online as supplementary material (Ayala-Rivera et al., 2023, 2024)). These templates have been designed considering an MVC-based web application. Reflecting this architectural pattern has involved defining and using three tokens in the pre-filled templates (i.e., view, service, and db tokens, corresponding to the three components of the MVC design pattern). Engineers, who are familiar with their system's architecture, should be able to easily customize the templates as needed. For instance, the only change required to adapt the pre-filled templates to a non-MVC application would be to define/use their own set of tokens and appropriate behavior. For example, if the application uses a Flux architecture (Boduch, 2016), it would require defining four tokens that reflect the four components of view, action, dispatcher, and store.

Listing 2 provides an example of the pre-filled template for the Web Filtering control (while Appendix presents all 10 pre-filled templates). This advice forwards all the request instructions to the WebFilter component. This component allows a request to proceed if the URL of the request is not in the blocklist, otherwise, it denies the request.

Listing 2: Pre-filled template for Web Filtering

```
For DPPs [6,7]
With new actors [WebFilter]
With sorting [WebFilter:0]
Pointcut composed of [4] messages involving the [request]
operations, the [$view,$service] actors, and the [URL,Connection]
data types.

Advice of type [REPLACE_ALL] with the instructions:
   $view -> WebFilter: getConnection()
   WebFilter -> $view: returnConnection()
   $view -> WebFilter: requestURL(url)
   WebFilter -> WebFilter: validateRequest(IP, url)
   alt if URL!=blackList
      WebFilter ->  $service:fwdRequest(URL)
      WebFilter <-  $service: response(URL, status)
      $view <- WebFilter: response(URL,status)
   else
     $view <- WebFilter: deny(URL)
     $view <- WebFilter: closeConnection()
```

4.2.4. Injection of Priv&Sec controls

SoCo injects the priv&sec controls automatically into the non-compliant diagrams. When done manually, this is one of the most time-consuming and error-prone tasks performed by engineers, and thus, fully automating it yields substantial benefits. First, our tool tries to match the pointcuts defined in the templates of the selected priv&sec controls against the non-compliant diagrams. Then, for each matched pointcut(s), our tool modifies the diagrams by injecting the corresponding advice. For demonstration purposes, assume that the Logging control (shown in Fig. 6(b)) needs to be integrated into the diagram shown in Fig. 1. This is a control that applies to *alteration* operations of *studentRecord* data types in the *database* so that logging instructions can be added after the pointcut.

Pointcut Matching: Initially, our tool performs fail-fast validations to discard irrelevant diagrams quickly. First, it checks whether the diagram includes the actors specified in the template associated with the control. Next, it checks whether the actors, operations and personal data items specified in the pointcut exist in the diagram. These checks are performed by matching entities between the templates and the diagrams. To promote the standardization and reusability of the templates, the values used to specify the controls must correspond to the *keywords* in the data dictionary. In contrast, the values found in the messages' diagrams can correspond to *related terms* (see Fig. 4). This strategy supports the interoperability of the components in SoCo. For example, if the pointcut defines the *"alter"* operation (a keyword in the data dictionary), all the terms related to this type of operation will also qualify as pointcuts, such as *"update"*, *"adapt"*, or *"modify"*. When our tool searches for that pointcut in the diagram shown in Fig. 1, two messages with the text *"updateStudentRecord(grades)"* qualify w.r.t. the operation in scope *alter* (the 2nd and 5th messages). Still, only one of them (the 5th message) has the actors involved (*"SIS → DB: updateStudentRecord(grades)"*), as it includes the DB as a message recipient.

Advice Injection: Then, SoCo applies the instructions specified by the advice. In our example, this involves adding the messages to support logging after the operations matching the pointcut are identified.

Post-processing: Finally, SoCo updates the look and feel of the evolved diagrams. It highlights in blue the messages added and in red the messages deleted or replaced. It also preserves the actors' order. Fig. 8 shows the evolved diagram of the described example.

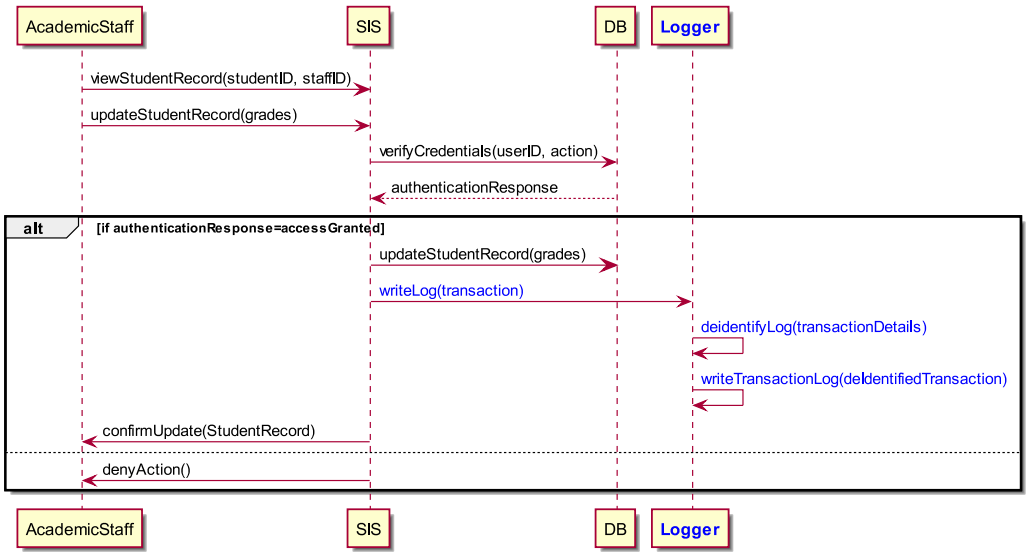


Fig. 8. Evolved Diagram for the example presented in Fig. 1.

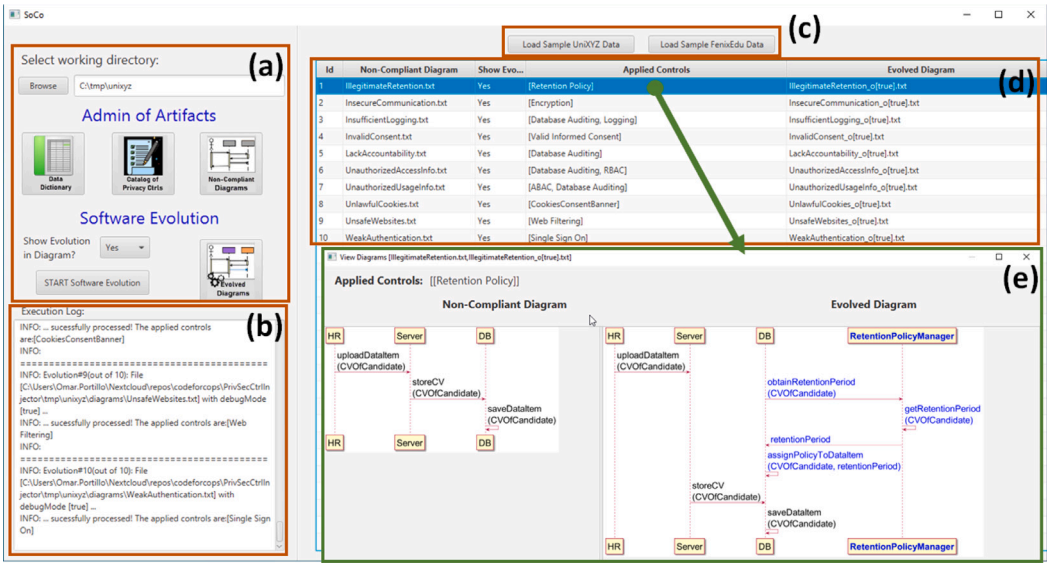


Fig. 9. GUI: (a) Configuration Panel, (b) Execution Log Panel, (c) Sample Data Buttons, (d) Results Panel, and (e) View Diagram Window.

5. Prototype

To prove the feasibility of SoCo, we implemented a tool that can be used by engineers to perform the steps of SoCo. We used PlantUML to represent the sequence diagrams and configure the priv&sec controls. It is also used to inject the priv&sec controls into the diagrams to evolve them.

Fig. 9 shows the main screen of our tool, which is composed of the following 5 main areas: (a) The *Configuration Panel* allows engineers to indicate the filepath where the inputs and outputs of SoCo will be located. The panel also allows engineers accessing and modifying the data dictionary, the catalog of priv&sec controls, and the set of non-compliant and evolved diagrams. For example, Fig. 10 depicts the window to manage the data dictionary specifying the keywords (key) and their associated related terms (value). Similarly, Fig. 11 shows the window to configure the priv&sec controls following the structure of the specified templates (i.e., capturing general information, and defining pointcuts, advice, and tokens). Fig. 12 depicts the window for exploring the non-compliant and evolved diagrams. It also illustrates data flow filters, as they are used to inspect the diagrams. Fig. 12 illustrates the creation of a data flow filter where the personal data item of *CV* has been selected and applied. Since only one diagram in the UniXYZ example satisfies the data filter (i.e., IllegitimateRetention), that is the diagram shown in the results table. (b) The *Execution Log Panel* displays relevant log information about the tool execution (e.g., progress status or errors – in case any of the inputs is invalid –). (c) The *Sample Data Buttons* allow engineers replicating the two example applications (i.e., UniXYZ and FenixEdu) discussed in this paper. After clicking one of the load buttons, the window shown in Fig. 13 will pop up. Through that window, engineers can replicate the execution of the selected example application by sequentially clicking the buttons of the four steps. Engineers can also see a summary of the tasks that each step involves on this screen. This window is not modal, so engineers can switch to other parts of the tool and better explore the intermediate results. (d) The *Results Panel* lists the resulting evolved diagrams, including the priv&sec controls that were applied. (e) The *View Diagram Panel* allows engineers double-clicking on a row to open a window that shows a side-by-side comparison of the non-compliant diagram and its evolved version.

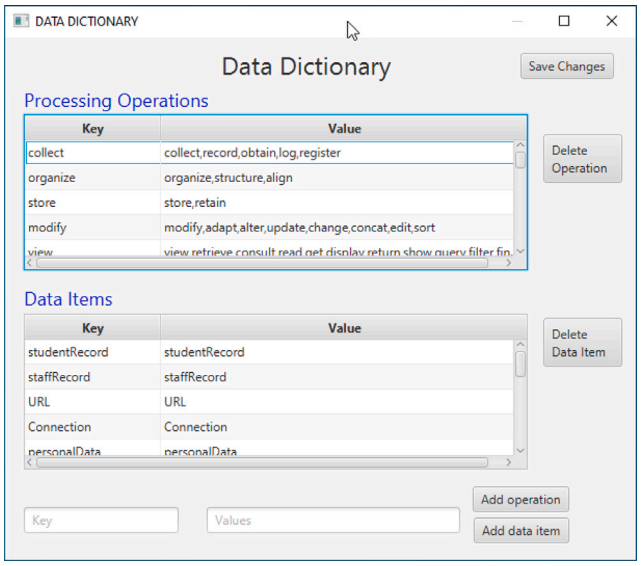


Fig. 10. Data Dictionary Window.

Due to space constraints, it is not possible to show screenshots of every single window in the tool. Nevertheless, it is worth noting that the tool supports Create, Read, Update, and Delete (CRUD) operations for all discussed components. For instance, engineers can use additional priv&sec controls to the 10 currently supported by SoCo out-of-the-box (i.e., those listed in Tables 2 and 3). To do this, engineers would only need to create and configure new control templates to model the desired controls.

We assessed the correctness of our prototype using 10 sequence diagrams based on the UniXYZ motivating example. Each diagram involves a personal data processing activity which is assumed to be non-compliant with a GDPR DPP and that could be fixed by integrating a suitable priv&sec control (based on our catalog). Table 4 summarizes this information. We also created a list of the minimal instructions (UML elements) necessary to apply each priv&sec control. With such instructions, we created the pre-filled templates (i.e., defined as a set of pointcuts and advices) of the 10 priv&sec controls in-scope for our experimental evaluation. We then verified whether the evolved diagram included the priv&sec control that we expected in the appropriate location. For the ten sequence diagrams, all expected controls were applied correctly. We originally assumed that each control would be applied once to each diagram. However, this assumption was incorrect for the Database Auditing control because this control was applied in 4 diagrams instead of 1 (i.e., diagrams #10, #1, #6, and #7). It is worth stressing that the purpose of this assessment was to act as a round of smoke testing (Nanayakkara et al., 2022), and helped us to determine that the prototype tool worked correctly and as expected. Thus, it was ready to be used in the formal evaluation of SoCo.

6. Evaluation: Case study

We aim to understand how a software engineer can use SoCo in practice to support GDPR compliance. We use a case study (Runeson et al., 2012) to assess the manual effort to apply SoCo to a large-scale software system. Our evaluation aims to answer two research questions: (RQ1) What manual activities a software engineer should perform to apply SoCo in practice to a large-scale software project? (RQ2) Can SoCo be applied to a large-scale software project?

Our unit of analysis is FenixEdu (Instituto Superior Tecnico, 2019), an open-source web application used in higher education institutions (HEIs). It is a large and complex application composed of ~445K lines of code, 228 entity types, and 343 processing activities of varying complexity (e.g., academic reporting and attendance tracking). We selected FenixEdu for our evaluation because of the following reasons: (1) Multiple studies have identified Education as a critical sector in the cybersecurity landscape whose security needs to be drastically improved after reporting a significant increase in cyber attacks (Ulven and Wangen, 2021; Pinheiro, 2020; Emeksiz and Zahadat, 2023; SonicWall, 2024). This is because universities store a large amount of sensitive data (such as research, financial, employment, medical, intellectual property, and tax) and research on cybersecurity risks in higher education is scarce. (2) An additional benefit of selecting an academic system is that we can leverage our personal knowledge and expertise in the area (i.e., as subject–matter experts) to understand the implications of security and privacy issues in Education; (3) FenixEdu is a real-world web application that is not only used by several Portuguese universities (like the Universidad de Lisboa, which is the largest in Portugal), but also by many other HEIs worldwide. Furthermore, the development community of this project is highly active with several organizations offering operational support and actively contributing to the application (Cachopo et al., 2011). These characteristics make FenixEdu a fair representative of the systems used in the HEI sector. (4) From a functional perspective, FenixEdu is a mature and robust application suite, composed of 12 different applications, with a broad functionality that covers far beyond a typical student information system and a learning management system. FenixEdu supports not only the academic management operations of a HEI (e.g., program, module, and assessment) but also the people management (e.g., students, staff, alumni) and the facilities management (e.g., classrooms). This characteristic allowed us to evaluate SoCo across a range of significantly different functional behaviors within a single application suite (i.e. FenixEdu).

6.1. Preliminaries

We asked a software architect with 15+ YoE (hereafter referred to as the participant) to apply SoCo to FenixEdu. Per the participant's self-assessment, he has expertise in web development. He is not a cybersecurity expert but has conducted GDPR awareness training. Regarding logistics, we agreed on a timeframe of four calendar months to complete this case study, providing on-demand support to the participant. We also asked him to keep a diary where he could document the time spent, and the activities performed in each step of SoCo and his impressions of using SoCo, with a particular emphasis on the challenges he faced. Regarding ethical considerations, we obtained informed consent (ICO UK, 2018a) from the participant before performing the study. We explained to the participant the aims of the study, the data that would be collected (e.g., his demographic information related to his professional experience, his behavior during the study, etc.), and the data to be disseminated (e.g., information to contextualize the case study results). Finally, the participant received a £100 Amazon gift card as compensation.

Pre-requisite Knowledge. Before engaging in the study, the participant was asked to read some introductory material about the GDPR DPPs and a description of the priv&sec controls in the catalog. He also invested 4 hrs in familiarizing himself with FenixEdu's functionality and architecture. This was necessary for him to be capable of making technical decisions (e.g., configuring the priv&sec controls according to FenixEdu's architecture). Finally, we delivered a 2-hr workshop to the participant to explain SoCo and how to use it for software evolution. The workshop involved running a test with the UniXYZ's diagrams, giving the participant an illustration of the pre-filled templates, and an introduction to the PlantUML syntax.

SoCo's Input: Data Dictionary. Since we did not perform an entire GDPR compliance exercise, we created a data inventory (i.e., artifact resultant of a data mapping activity) using two publicly-available lists of personal data collected by universities (UCD, 2024; Maynooth University, 2018). This resulted in the documentation of 48 personal data items (e.g., email, salary) and 45 processing operations (e.g., store,

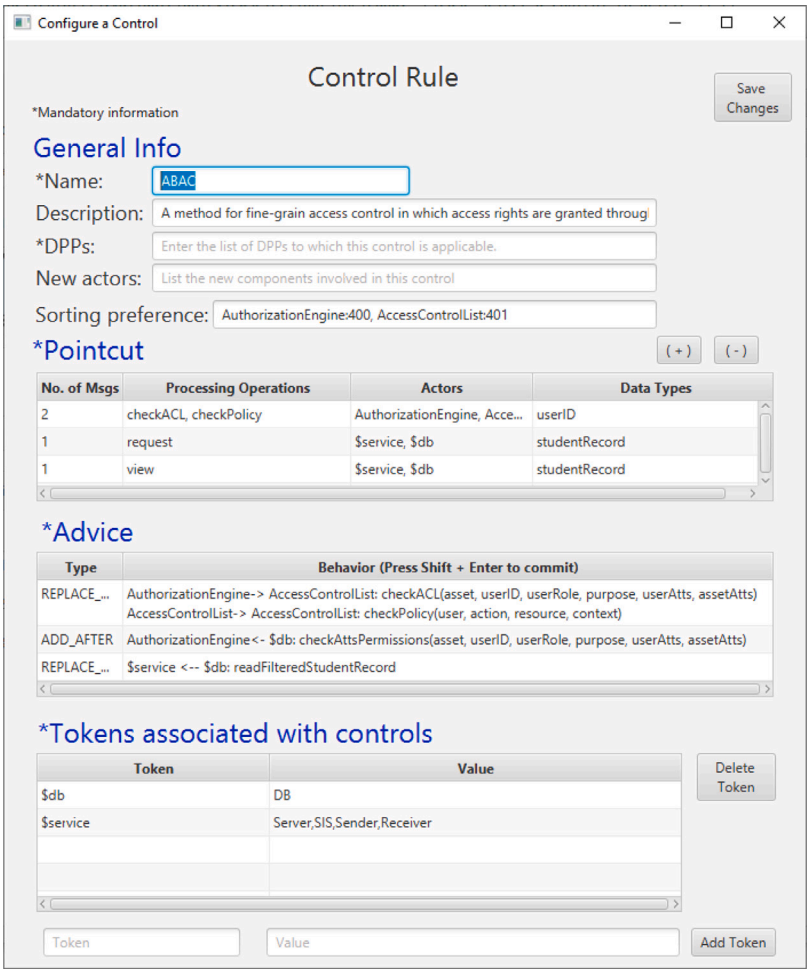


Fig. 11. Priv&Sec Control Configuration Window.

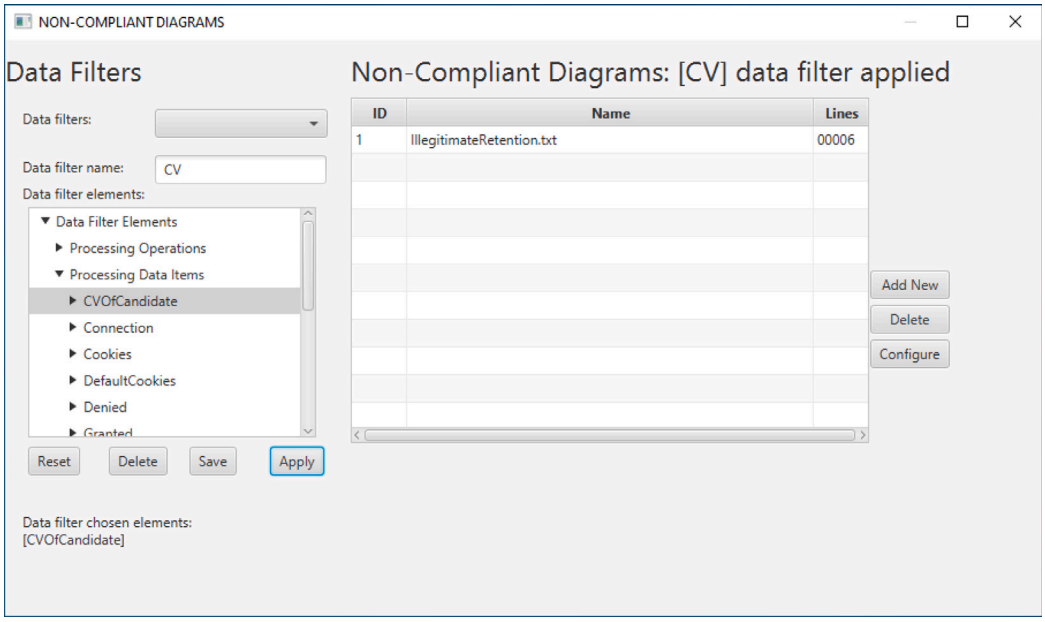


Fig. 12. Non-Compliant/Evolved Diagrams Window.

alter). Based on such data inventory, we created the data dictionary. The data inventory's personal data elements became the keywords, while their respective synonyms became the related terms. This step is necessary to expand the data filter search space and mitigate the impact of possible word inconsistencies in the sequence diagrams. We obtained the synonyms from Wikipedia via the Word2vec tool (Yilmaz

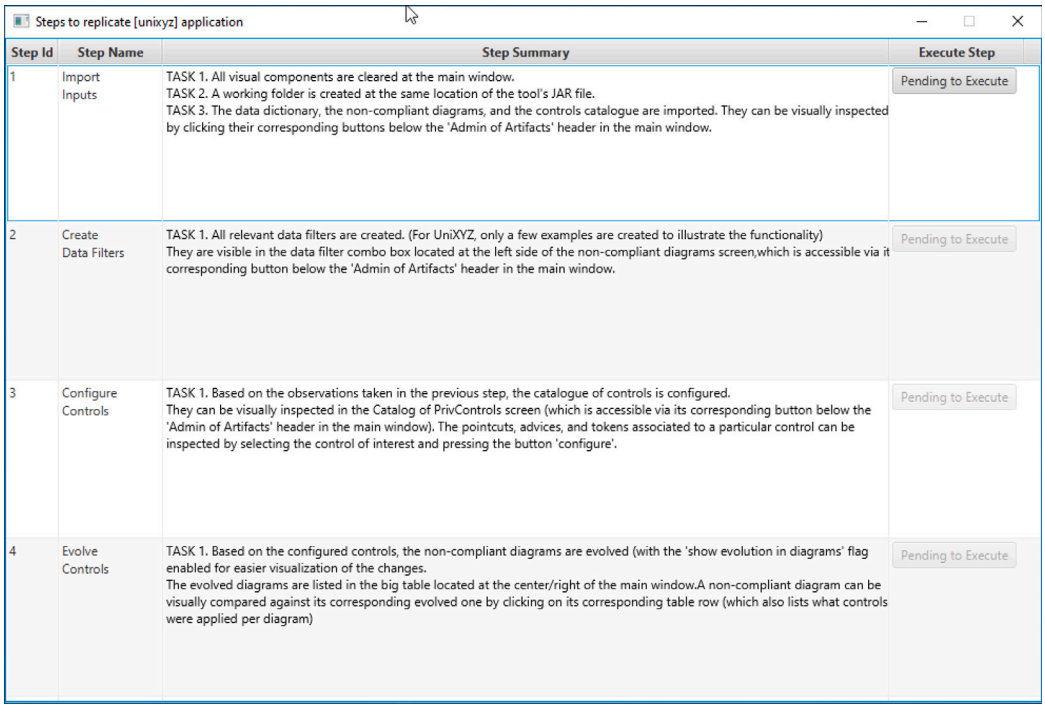


Fig. 13. Sample Data Window.

Table 4
Diagrams, GDPR DPPs, and controls used in UniXYZ study.

#	Non-compliance issue	DPP violated	Recommended control
1	Lack of accountability mechanisms	Accountability	Database auditing
2	Access to malicious and unsafe websites	Integrity and Confidentiality	Web filtering
3	Invalid consent management	Purpose Limitation	Valid informed consent
4	Insecure communication	Integrity and Confidentiality	Encryption
5	Unlawful cookie management	Lawfulness, Fairness and Transparency	Compliant cookie consent banner
6	Unauthorized access to sensitive information	Lawfulness, Fairness and Transparency	Role-based access control (RBAC)
7	Unauthorized usage of sensitive information	Data Minimization	Attribute-based access control (ABAC)
8	Illegitimate data retention	Storage Limitation	Retention policy
9	Weak authentication	Integrity and Confidentiality	Single Sign-On
10	Insufficient logging	Accuracy	Logging

and Toklu, 2020; Jatnika et al., 2019). We set 3 as the maximum number of synonyms to retrieve per data inventory's keyword to limit Word2vec's search space to the most relevant associated terms.

SoCo's Input: Non-compliant diagrams. The design documentation and artifacts for FenixEdu are not available. Thus, we applied reverse engineering over FenixEdu's source code using ObjectAid (ObjectAid, 2022) and a converter from ObjectAid to PlantUML (PlantUML, 2023) to generate the UML sequence diagrams in the required syntax for SoCo. This resulted in 343 sequence diagrams. However, we are interested in those that are GDPR-relevant, that is, those diagrams that involve processing personal data. Therefore, we created a script to identify those diagrams automatically. To recognize GDPR-relevant interactions, the script iterates over the data dictionary's keywords and tries to match them against the names of classes, methods, and parameters in the diagrams. We assume that using these matching elements can suffice for finding relevant diagrams since engineers usually follow meaningful programming naming conventions (e.g., verbs, nouns) (Butler et al., 2015). A diagram qualifies as GDPR-relevant if at least one message in the diagram contains one of the following combinations of keywords or their related terms: (1) a processing operation + a personal data item (e.g., store + curriculum vitae), or (2) a processing purpose + a personal data item (e.g., advertising + employment status), or (3) a processing operation + a processing purpose (e.g., modify + appeal examination). We used "contains" (ignoring case) as the main function to compare the text inside the diagrams. This resulted in 285 diagrams identified as GDPR-relevant. The relevant diagrams involved mostly student-centric data flows such as students' grading. The non-relevant diagrams involved administrative processing activities (e.g., the configuration of holidays), while other diagrams did not qualify as relevant because the method's name was in Portuguese instead of English (e.g., *setDisciplinaExecucao*). Finally, for our study, we assumed that all the relevant diagrams were non-compliant with the GDPR DPPs. The rationale behind this assumption is that FenixEdu was not designed and developed as a GDPR-compliant application.

6.2. Manual activities

This section discusses the activities performed by the software engineer to apply SoCo in practice (RQ1).

6.2.1. Generation of data flow filters and selection of Priv&Sec controls

The participant carried out the generation of data flow filters and the selection of priv&sec controls in parallel.

He reviewed the diagrams (one at a time) to determine which priv&sec controls (among those shown in Tables 2 and 3) would make each diagram compliant. He also noted how to configure the pointcuts in the priv&sec templates based on the elements observed in the diagrams (e.g., what messages and actors to consider implementing a priv&sec control in a diagram). In this part, the participant pointed out that FenixEdu was a well-structured application (e.g., using naming conventions and having well-defined classes), which facilitated the identification of the pointcuts as the more diagrams the participant examined, the more similarities he found among the non-compliances and their pointcuts. For instance, all database access occurs through the *domain* package in FenixEdu's code. This facilitated the identification of database-related messages, which are relevant for priv&sec controls like Audit and Logging. The participant sorted the diagrams from the least to the most complex to gradually "warm-up" by working on the most straightforward diagrams first. Complexity was measured in terms of the number of messages in the diagram.

Whenever the participant identified a new pattern for a potential pointcut in a diagram (i.e., the possible location where a priv&sec control could be applicable), he created a data flow filter to find similar diagrams. For example, the participant created filter identifying CRUD operations to quickly retrieve the diagrams where an Audit control should be applied. Overall, he created 31 filters and assigned them meaningful names to recognize them later. The participant adopted a naming convention for the data flow filters starting the name with a prefix based on the control that he was analyzing (e.g., log for logging), followed by the list of data filter elements separated by "_" (e.g., log_create_db_alumni).

6.2.2. Configuration of Priv&Sec controls

The participant configured the priv&sec controls' templates to fit FenixEdu's application context. This activity was straightforward for the following reasons: 1. It was only necessary to apply minor changes to the pre-filled priv&sec control templates. The only change for 7 (out of 10) templates involved modifying the tokens associated with the actors according to the system architecture components. For example, FenixEdu has 200+ classes related to database access components. By using the token $*db* to refer to these classes, it was possible to capture all the database components in FenixEdu and share this value across all the templates. The capabilities of supporting multiple values per token, and sharing the tokens among controls, were helpful to keep the number of tokens low (only four were needed) and avoided using more than one template per priv&sec control. The other 3 (out of 10) templates required customizations to their advice behavior to fit FenixEdu's architecture. For instance, the RBAC and ABAC control templates needed some adjustments to their instructions to interface with FenixEdu's existing authorization logic to use it as a proxy and avoid changing other system parts. Still, the participant pointed out that modifying the instructions was straightforward thanks to the intuitiveness of PlantUML's syntax and the expressiveness of our pointcut and advice language. 2. FenixEdu's architecture follows good practices (e.g., the MVC design pattern), which facilitated defining the pointcut patterns. In a poorly-designed system, more effort would be needed to configure the templates (e.g., more instances of the templates per priv&sec control, more complex pointcuts/advices). 3. The fail-fast validations of our tool informed the participant about misconfigurations, pinpointing what needed fixing (hence avoiding wasting time). For example, the tool notifies if a template is not applied to any diagrams (suggesting an omission) or if a template uses nonexistent entities (suggesting the presence of a typo in the configuration of the template).

6.2.3. Injection of Priv&Sec controls

The participant used SoCo to inject the priv&sec controls into the 285 non-compliant diagrams automatically. Then, he assessed the correctness of the evolved diagrams manually. He compared each evolved diagram against the original one to verify whether our tool recognized the pointcuts specified and applied the advices in the correct location. This task was significantly time-consuming, taking 95 hrs. We documented the number of times each control was injected into any of the FenixEdu diagrams (see Table 5). Database Auditing and Web Filtering were the two most common controls applied in FenixEdu, followed by Encryption and Single Sign-On. Database Auditing was applied when the diagram involved any CRUD operation of personal data to the database of FenixEdu. Web Filtering was applied to diagrams involving web requests from the view to the backend. Since FenixEdu is a web-based application, this was a prevalent scenario in the system. It is worth noting that the broad range of functionality offered by FenixEdu, which has (functionally) evolved beyond a traditional SIS to cover most of the supporting functions typically required by an HEI, allowed us to try multiple different functional behaviors within a single case study. To better illustrate this point, a breakdown of the number of times each control was injected into any of the FenixEdu diagrams clustered per functional module is presented in Tables 6 and 7. Table 6 presents the information of the functionality typically found in a SIS (e.g., programme/module/assessment management), while Table 7 presents the information of the functionality which is not typically part of a SIS (e.g., facilities and accounting management).

The participant mentioned that SoCo helped fix confirmed GDPR violating scenarios, discover flows of personal data in the system, and improve the overall system security. This is because he only had to configure patterns of personal data processing operations along with the data elements involved, and SoCo would find all the scenarios that match those patterns to inject applicable security measures.

The results showed that most controls were applied correctly, with no pointcut instances omitted. The injection logic worked correctly for 81.5% of the cases (i.e., 661) and incorrectly for 19.5% of the cases (i.e., 160). The error cases exhibited an issue where the text of a message was modified incorrectly by our tool while injecting the priv&sec controls. This happened when the following conditions co-occurred: (1) The user enabled the option to highlight in color the changes made in the evolved diagrams, (2) two controls were applied to the same diagram, and their advices had the same messages, and (3) the

Table 5
No. of times a control was applied to FenixEdu.

ID	Priv&Sec control name	No. of times applied
1	Database Auditing	245
2	Web Filtering	285
3	Valid Informed Consent	5
4	Encryption	80
5	Compliant Cookies Consent Banner	5
6	RBAC	59
7	ABAC	23
8	Retention Policy	7
9	Single Sign-On	80
10	Logging	32
	TOTAL:	821

Table 6
No. of times a control was applied to FenixEdu's core SIS functionalities.

ID	Student Mgmt	Staff Mgmt	Programme Mgmt	Module Mgmt	Assessment Mgmt	Thesis Mgmt
1	41	22	23	65	10	17
2	45	24	25	73	10	20
3	0	0	0	0	0	1
4	9	15	16	18	2	7
5	4	1	0	0	0	0
6	6	12	12	17	0	1
7	3	3	3	9	3	0
8	2	0	0	0	1	2
9	9	15	16	18	2	7
10	8	3	3	0	2	0
TOTAL:	127	95	98	200	30	55

Table 7
No. of times a control was applied to FenixEdu's non-SIS functionalities.

ID	User Mgmt	Accounting Mgment	Website Mgmt	Alumni Mgmt	Facilities Mgmt	Project Mgmt
1	5	17	8	15	2	20
2	5	19	15	19	4	26
3	0	0	1	3	0	0
4	1	9	0	0	0	3
5	0	0	0	0	0	0
6	0	8	0	0	0	3
7	0	2	0	0	0	0
8	0	1	0	1	0	0
9	1	9	0	0	0	3
10	1	7	3	2	0	3
TOTAL:	13	72	27	40	6	58

Table 8
Effort comparison: UniXYZ and FenixEdu.

Effort category	UniXYZ (mins)	FenixEdu (mins)	FenixEdu/ UniXYZ (ratio)
Controls Sel. & DF Gen. time (manual)	20.0	4944.0	247
Controls config. time (manual)	5.0	30.0	5
Controls injection time (tool)	0.2	13.1	65

advice of the second control involved replacing some of the messages previously modified by the first control. Regarding the implications of this issue, the controls were injected in the right location, and the correct set of messages and actors were modified, removed, and/or added . However, the text of the messages that overlapped (i.e., those previously modified by the first control) was corrupted, as the color tag added to represent its previous modification was not removed. For instance, the text "color blue" should not be duplicated in the following message *"AccountingEventsCreator → AccessControl: <color blue> [color blue]getPerson() : Person"*. This issue impacted all the injections involving the SSO and Encryption controls (as both controls were applied to the Person entity).

6.3. Applicability to a large-scale software project

To evaluate SoCo's applicability to FenixEdu (RQ2), we evaluated its performance, overhead, and time savings compared to a completely manual approach.

6.3.1. Performance

We assessed the time needed by the participant to apply SoCo to FenixEdu and compared it with the time required to apply SoCo to our toy example, UniXYZ. Table 9 presents the complexity information of both applications. We classified diagrams into three categories based on the number of actors and messages: *small*, *medium* and *large*. Regardless of the classification criterion, UniXYZ only has *small* diagrams, whereas FenixEdu has diagrams across all three categories. We measured the execution time reported in the various steps of our approach (see Table 8). Due to its complexity, the time to perform all the steps is always higher when SoCo is applied to FenixEdu.

By far, the analysis of the diagrams to determine which controls need to be selected involved the highest ratio. This effort depends on the number of diagrams (whose FenixEdu/UniXYZ ratio is 28) and the number of messages. On the contrary, configuring the templates of the priv&sec controls in FenixEdu required only five times the effort we spent with UniXYZ. This resulted from using our pre-filled templates, which only required the participant to perform some adjustments to tailor them to FenixEdu. The time to inject priv&sec controls was also higher in FenixEdu due to the number of non-compliant diagrams (285, while UniXYZ had 10) and their size (the average number of messages per diagram was 59, while in UniXYZ was 8). However, the throughput of our tool (i.e., the number of messages processed per second) was the same in both cases (ten messages). This suggests that injection scales linearly w.r.t. the total number of messages, irrespectively of the actual number of diagrams.

6.3.2. Overhead

We also studied the overhead of using SoCo to complement our analysis. In addition to the execution time (previously discussed), our other main metrics were CPU (%) and memory (MB) utilization, as well as the prototype's throughput (i.e., number of messages processed by second). Garbage collection (GC) was also monitored, as it is an important performance concern in Java (Portillo-Dominguez et al., 2016) (which is the technology used to develop our prototype). The prototype was run in a computer running Windows 10 64-bit, with 32 GB of RAM and an Intel i7 CPU at 3 GHz. The Java version used was Hotspot JVM 1.19 (with a heap of 2 GB) and JavaFX 1.19.

Table 10 summarizes the data collected: The average and maximum CPU/memory utilization, the time spent on GC, and SoCo's throughput per second (TPS) for both UniXYZ and FenixEdu applications. SoCo demonstrated to be lightweight in terms of CPU. The average CPU utilization did not exceed 15% in either case (despite the significant difference in complexity among the two applications). On the contrary, the average memory consumption was significantly higher for FenixEdu w.r.t. UniXYZ (approximately 200MB). This is explained by the fact that the input required by FenixEdu (mainly the diagrams) is considerably higher (as FenixEdu is composed of 285 diagrams, in contrast to UniXYZ, which only has 10 diagrams). The prototype's current memory footprint is relatively high (as most of the memory reported by UniXYZ belongs to the prototype itself). This is explained by the fact that we have not conducted any performance optimizations for the current prototype. Nevertheless, SoCo could process FenixEdu without issues with the assigned memory (i.e., the Java heap). Overall, CPU and memory usages were considered tolerable because the computer was far from exhausting its resources. Furthermore, no significant GC overhead was observed, as the total experienced GC in both cases was less than 1% of the total execution time (another strong indicator that the memory settings were appropriate). Finally, the throughput of our tool was the same in both cases (10 messages). This suggests that the injection of controls scales linearly w.r.t. the total number of messages (irrespective of the actual number of diagrams).

6.3.3. Time savings

We also analyzed the possible time savings gained by using SoCo compared to performing the same tasks manually and without any tool support. To do this, we have focused on estimating it for SoCo's step #4 (i.e., the injection of priv&sec controls) because this task has been fully automated; hence, it has the highest time-saving potential within SoCo.

FenixEdu: To do this analysis, we estimated the time necessary to apply the software evolution activity to FenixEdu's manually, based on two assumptions: (1) We assumed a linear growth in the effort required along with the complexity category (i.e., the *medium* and *large* complexities requiring 2 and 3 times, respectively, the effort of the *small* category). (2) We assumed that evolving one diagram of the *small* category in FenixEdu required the same effort that doing it in UniXYZ. This second assumption allowed us to leverage the available UniXYZ's information, which included the time used to conduct it manually by us. In UniXYZ, performing the manual injection task took 23 min in total (significantly higher than the 0.2 min taken by SoCo), for an average effort of 2.3 min per diagram. This information allowed us to calculate an estimated time saving for FenixEdu, gained by using SoCo, in the range of 23–27 hrs (considering the message- and actor-based classifications, respectively).

It is important to highlight that the above analysis is a *conservative* estimate. This is because the manual tasks in UniXYZ were performed by us, the authors, who are practitioners with significant experience in both the software development and privacy and security fields. Thus,

Table 9
Actor-/Message-based complexity categories.

Application	Category w.r.t. No. of actors			Category w.r.t. No. of msgs		
	Small (02–05)	Medium (06–10)	Large (11+)	Small (02–30)	Medium (31–60)	Large (61+)
UniXYZ	10	0	0	10	0	0
FenixEdu	32	93	160	81	96	108

Table 10
SoCo's resource utilization.

Application	CPU (%)			Memory (MB)			Others	
	Avg.	StdDev.	Max.	Avg.	StdDev.	Max.	GC (s)	Throughput (mps)
UniXYZ	3	6	19	469	195	1075	0.3	10
FenixEdu	14	2	21	678	281	1230	17.5	10

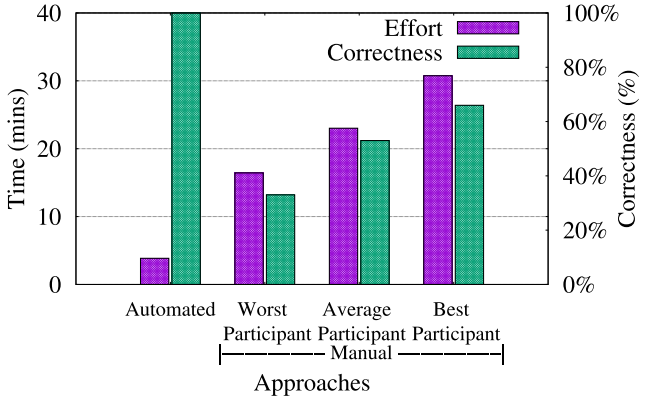


Fig. 14. Effort/Correctness comparison per approach type.

it is reasonable to expect that the manual injection would have taken longer for less experienced practitioners. Likewise, as FenixEdu is far more complex than UniXYZ, even its *small* diagrams are considerably larger and more complex than their UniXYZ's counterparts. Thus, it is reasonable to expect that the manual injection in FenixEdu should take longer. Finally, it is worth mentioning that the current prototype tool is in its early stages of development and lacks any sort of performance tuning such as parallel computing optimizations, which would certainly decrease the execution time of the tool (hence, increasing the time savings of using it).

UniXYZ: To offer a complementary perspective of the time savings that SoCo can achieve, we conducted an additional exercise with six Computer Science Ph.D. students (hereafter referred to as the students) with significant (5+) years of experience in software development, as well as practical experience in cybersecurity and GDPR awareness. We asked students to modify manually the 10 UniXYZ sequence diagrams (listed in Table 4), by designing a given priv&sec control (see 3rd column in Table 4) to address the existing compliance issue. That is, students had to modify (as needed) the sequence diagrams representing the non-compliant scenarios by adding/modifying/removing messages, parameters, actors, and/or objects. Although for each diagram we gave the students the name of the control to integrate, we did not indicate how, when, or where to inject the control within the diagram.

We then compared the effort (in minutes) required to integrate a priv&sec control in the diagram using SoCo against doing it manually. Fig. 14 compares the time spent by the students (showing the best, worst, and average performer) versus SoCo (whose time includes the configuration of the controls' templates and the tool's execution time). When compared against the manual approach, SoCo ensured significant time savings (between 77% and 83%) without sacrificing correctness, as it integrates the priv&sec controls as specified in the template (whose configuration for such simple diagrams involved minimal changes, and took most of the time reported by SoCo in Fig. 14). It is important to highlight that the time savings achieved were considerably relevant, despite the fact that the number of diagrams was small (only 10) and their complexity was low (as reported by Table 9).

We also evaluated the correctness of the modifications applied by the students. For that purpose, we compared the modified diagrams against a list of the (minimal) instructions necessary to implement each control, and assigned points (using a 4-point scale) to each diagram depending on the number of changes that were covered. The percentage of satisfactory adaptations across the 10 diagrams for all students ranged between 33% and 63% (as seen in Fig. 14), where the automated approach (i.e., SoCo) was considered the ground truth with 100%. This result illustrates the challenges typically faced by engineers to manually conduct these types of tasks as, despite the low complexity of the UniXYZ diagrams (as shown in Table 9), not all the modifications done by the students were satisfactory. The most common error observed across the students was the incomplete design of a control. For example, in diagram #7, 5 (out of the 6 students) did not consider the user and resource parameters when designing an access control function to enable finer-grained access rights. The most difficult diagrams, as per our results analysis, were #6 and #7. This is because both RBAC and ABAC required multiple attributes (e.g., policies and roles), which were difficult not to overlook when evolving the diagram. On the contrary, the easiest diagram was #4, as encryption is a control commonly applied in software systems, which is also relatively easy to design. Thus, most of the students were familiar with it.

6.4. Additional observations

The participant also provided suggestions to improve SoCo. They are discussed in the following paragraphs:

Word2vec: Using Word2vec helped complement the data dictionary terms but can also bring the risk of retrieving wrong terms. For instance, for keyword *formation*, Word2vec retrieved the terms *for, form, formation*. However, *for* (preposition, conjunction) is a prefix of multiple words. Thus, if this was used to create data flow filters, it caused to retrieve scenarios that are irrelevant to the desired search. The issue can be addressed by validating the synonyms manually.

Controls Templates: The participant preferred to define pointcuts composed of the least possible number of messages and mainly used the REPLACE and ADD advice types to reduce the impact of the changes to the existing system architecture. Even though the advice instructions

are simple, these still offered sound guidance to the participant about what controls to integrate into a particular data flow scenario and where, as developers will determine the how. Using simple instructions for pointcuts is feasible due to FenixEdu's well-designed architecture (i.e., modular and loosely coupled MVC) and it may not have been possible in a poorly-designed system.

Data Flow Filters: The participant recommended using additional types of operators to define data filters (e.g., arithmetic and logical). He also mentioned that supporting a SQL-like language would allow the creation of more flexible search criteria. Tokens proved to be quite beneficial in looking for patterns involving data not found in the data dictionary. However, supporting wildcards to create even more flexible search criteria would be helpful.

Syntactic Text Matching: Using a syntactic matching approach had some limitations in the data flow filters. For instance, when the participant included the processing operation *add* as part of a filter, the tool also returned diagrams where this string was present within a message, such as an uplo*add*ataItem, which was incorrect. As a workaround, the participant ignored those diagrams or refined the definition of the filter.

Diagram Visualization: The participant recommended implementing a zoom-in/out function to visualize diagrams that were too big. Another feature that would facilitate the review of the evolved diagrams is to distinguish what controls were applied and what changes were made as part of each control. Regarding color highlighting, the participant mentioned that the mixture of green (GDPR-relevant), blue (added), and red (removed) helped assess the correctness of the evolved diagrams.

Injection Process: The participant pointed out that the results of the injection process can only be explored once the whole process has finished. Thus, one recommendation was to display the evolved diagrams gradually in the results panel as they are completed, so the user can start their exploration sooner. Another suggestion was to have a "load evolved diagrams" feature in the tool, which allows retrieving the evolved scenarios from a directory and showing them in the results panel for exploration.

7. Discussion and future work

(a) Identification of GDPR violation scenarios: Organizations can leverage diverse types of strategies to identify GDPR violations such as studying guidelines (ICO UK, 2018b) or case studies (DPC Ireland, 2023) from supervisory authorities, conducting Data Protection Impact Assessments (DPIAs) (DPC Ireland, 2018), using threat modeling methodologies for security (Thevarmannil, 2023) or privacy (KU Leuven, 2023), or simply consulting the expert judgment from a legal team. SoCo has been purposely designed to be agnostic of such strategies, SoCo simply requires the non-compliant scenarios to be fed modeled as sequence diagrams. However, we acknowledge that integrating SoCo to such types of techniques can bring interesting benefits for practitioners (e.g., time savings, elimination of potential redundant steps). Therefore, finding effective ways to properly support artifacts derived from those techniques will be considered as a potential extension of our research work in the future.

(b) Priv&Sec Controls Evaluation Sample: Although only 10 priv&sec controls were part of our evaluation sample, our results demonstrate that they were sufficient to address the scenarios identified as non-compliant with GDPR DPPs in FenixEdu. This set of controls was selected because they are considered basic and foundational cybersecurity measures in organizations. Also, we assumed that security incidents involving a lack of robust priv&sec controls or improper implementation of these is what often lead to a data breach. Moreover, the controls selected were reviewed by a set of priv&sec experts and confirmed to help with GDPR compliance (as part of our previous work (Ayala-Rivera and Pasquale, 2018)). That said, SoCo is not restricted to that set of 10 priv&sec controls. More controls can be easily added to SoCo via the prototype tool (as discussed in Section 5). In the future, we plan to gradually extend the set of available ready-to-use (i.e., pre-filled) controls' templates to cover other software-relevant priv&sec controls recommended by security frameworks such as the Center for Internet Security (CIS) (CIS, 2023), NIST SP 800-53 (NIST, 2020), or ISO/IEC 27002:2022 (ISO/IEC, 2022b). Likewise, although our evaluation mainly involves security controls rather than privacy controls, our catalog and the frameworks considered expanding our work already integrate privacy controls.

(c) Case Studies and Participants' Profile: So far, SoCo has been evaluated by a seasoned software architect to demonstrate the feasibility of the approach and tools for supporting the software compliance with GDPR. Nevertheless, we plan to strengthen our evaluation by conducting more case studies, preferable involving multiple IT professionals and applications in other industry sectors, in order to further test our approach's performance and generalizability. To scale our evaluation to more participants, we consider that we need to improve the level of automation of some tasks in our tool (such as the verification of the correctness of the evolved diagrams), so that participants can use the tool more effectively. In this direction, we will also explore other ways to detect GDPR-relevant diagrams, for example, by leveraging threat-focused approaches (from privacy engineering) to detect data processing flows at risk automatically.

(d) Text Similarity Techniques: SoCo currently uses syntactic similarity to search for data items and operations in the sequence diagrams. We decided to go in this direction because engineers usually follow meaningful programming naming conventions, including using standard nouns and verbs for the entities and behaviors represented in a software system (Butler et al., 2015). Moreover, syntactic and lexical similarity techniques are commonly used in code scanning tools as they focus on the structural and grammatical aspects of code rather than its semantic or functional behavior (Gali et al., 2019; Kılıç and Sandıkkaya, 2023). Since the sequence diagrams fed to SoCo will ultimately be translated into code instructions by engineers, we followed this same principle. Additionally, we opted for not using semantic similarity techniques because words and phrases often have multiple meanings in natural language depending on the context, and automatic semantic matching might struggle to disambiguate terms correctly (hence, potentially resulting in false matches). Therefore, we opted to leverage semantic similarity to expand the data dictionary, and then review/clean it by removing any invalid terms/synonyms (as explained in Section 4). This strategy helps to achieve the benefits of semantic matching, while also preventing its potential issues (as discussed in our evaluation). Finally, although SoCo currently uses "contains" (ignoring case) as its primary function to assess similarity, it is planned to include other functions in the future (e.g., a semantic one).

(e) Translation of Sequence Diagrams to Code: Once engineers complete the steps involved in SoCo, the evolved sequence diagrams modeling the fixed GDPR DPP violations are ready to guide engineers in the development phase of the SDLC. That is, engineers would implement the security measures and associated changes suggested by SoCo to reflect the software evolution. Since the aim of SoCo is to support engineers in the design phase of the SDLC, SoCo currently does not offer a functionality to automatically generate source code based on the evolved sequence diagrams. However, automatic code generation from UML sequence diagrams is a well-studied field not only in the literature (Parada et al., 2011; Kundu et al., 2013; Paolone et al., 2020) but also applied in commercial tools (Altova GmbH, 2023). Depending on the programming language used, there are typically tools that can assist engineers with this task. For instance, integrated development environments (e.g., Eclipse, IntelliJ IDEA) usually offer UML modeling and code generation plugins. We plan to leverage some of these approaches to integrate them into SoCo to further automate the process of implementing priv&sec controls for GDPR compliance. Although the generated code may lack all the architectural considerations and optimizations that engineers might introduce (depending

on the complexity of the diagrams), this approach is common in the industry, as it provides a fair starting point for engineers to work with.

(f) Potential time-savings of SoCo: In our evaluation, all participants had a significant level of professional experience (i.e., at least 5 years). As the results have shown, the obtained time savings are evident and relevant, and so it is expected that SoCo can achieve better results when used by more inexperienced practitioners. Likewise, the potential time savings that SoCo can yield are directly related to the number and the level of complexity of the non-compliant diagrams. Therefore, the larger the number or diagrams is (and/or the higher their complexity), the manual effort required will also be larger. Hence, offering a lot of potential savings which SoCo can convert into actual time savings.

(g) Potential value of SoCo for researchers and practitioners: This research paper proposes practical artifacts and novel techniques for engineers to help them operationalize GDPR in software systems and improve their security and privacy. It also presents valuable insights and findings that can benefit researchers and practitioners in the field of secure software engineering. Next, we briefly discuss the main ones:

- *Practical guidance towards GDPR compliance*: Previous research work has identified that developers are not able to implement GDPR effectively into software development practice due to the lack of familiarity with GDPR DPPs and the lack of knowledge on what techniques to use (or how to implement them) (Alhazmi and Arachchilage, 2021). Our research provides value to practitioners as it helps to solve these issues by providing them with direct guidance on how to operationalize the GDPR DPPs in software systems. Through the use of our catalog that maps priv&sec controls with GDPR DPPs, practitioners can know what controls to use to fix a GDPR-violating scenario. Moreover, our proposed techniques support practitioners to know where in the data processing activity such priv&sec controls can be integrated and how they can be technically implemented. Practitioners can also gain a deeper understanding of where GDPR violations can arise in data processing activities present in software functionality and how they can be addressed because SoCo allows to filter sequence diagrams in the scope of GDPR. All these contributions help to advance the body of knowledge intersecting the fields of software security and software engineering.
- *Benchmarking Opportunities*: To promote research reproducibility, our research work includes a ready-to-use prototype implementation of SoCo which is publicly available as a GitHub project (Ayala-Rivera et al., 2023). This also includes easy-to-follow instructions to build the prototype, as well as an automated way to replicate the experiments performed in this paper (with only 5 mouse clicks). Other artifacts (such as the catalog of controls, scenarios used, etc.) are also available for researchers (Ayala-Rivera et al., 2024). We also plan to adopt FAIR principles (Lamprecht et al., 2020) for our tool and other artifacts developed in our research work. Thus, SoCo can act as a benchmark or rival technique for evaluating other research efforts tackling similar research problems. This can also facilitate comparative studies and foster healthy competition within the research community.
- *Support Automate Compliance Activities*: Our proposed approach can also contribute to the efforts of organizations to automate compliance activities. This can lead to more efficient and effective software development processes, ultimately helping to make software more secure and compliant, while also less costly and more maintainable. In particular, providing practical tools to support engineers in implementing priv&sec controls can help to increase the productivity of practitioners and at the same time reduce the level of errors that may lead to security incidents and therefore data breaches.
- *Inspiration for Future Work*: In this section, we have presented various ideas on how to expand our work. This can serve as a source of inspiration for future research projects. Researchers may identify areas that need further investigation, or explore some of the explicitly described pointers of future work in this area.

8. Threats to validity

In this Section, we discuss the main factors that may affect the validity (Runeson and Höst, 2009) of our results. Our evaluation relied on human judgment to verify the correctness of the evolved diagrams. This can affect the validity of our results due to observation bias. To reduce it, we recruited only participants with significant experience in software development. Although we considered only one application suite to evaluate our approach, it covers 12 different application (i.e., functional) modules which allowed us to diversify, to a certain extent, the evaluation of SoCo to a range of different functionalities. Also, the 285 diagrams that we considered covered all the types of GDPR DPPs violations and use the whole set of 10 in-scope priv&sec controls. In future work, we will apply SoCo to applications from domains different from a university information system.

Our evaluation involves one case study with a single participant. We acknowledge that having multiple case studies would help to have a higher validity than a single case study (allowing us not only to have better justified findings, but also probably other findings) (Bratthall and Jørgensen, 2002). Likewise, it would be desirable to have multiple participants in order to be more representative of real-life practitioners conditions, as professional software development is commonly a team activity. Unfortunately, research works have also reported the multiple challenges that might prevent researchers (like ourselves) to conduct more experiments/case studies with human participants (Buse et al., 2011; Ko et al., 2015). Nevertheless, the results obtained still offered valuable insights about SoCo's effectiveness, the effort required to use it, and the time savings SoCo can achieve. Similarly, relying on human expert judgment has the intrinsic risk of overlooking errors, especially when it involves manual inspection. To mitigate it, we exclusively recruited participants with significant professional experience in the software industry. To further address this risk, we plan to conduct more studies that involve multiple participants and explore ways to automate the evaluation of the correctness of the evolved diagrams.

The participants in our evaluation could have misinterpreted the GDPR DPPs and/or the priv&sec controls used. This can undermine the validity of our results. To mitigate this, we provided the descriptions from the GuideMe approach (Ayala-Rivera and Pasquale, 2018), which explain the DPPs and the priv&sec controls in layman's terms, to the participants before the studies were conducted. The order of the diagrams may have also impacted the time measures due to the learning curve of the participants. In the case study, the diagrams were ordered intentionally by their complexity to reduce the learning curve.

To identify the GDPR-relevant diagrams, we used Word2vec trained with Wikipedia to retrieve the synonyms of the data inventory's entities, as it has proved to have a broader coverage of terms than other corpora (e.g., Wordnet) (Muller and Gurevych, 2009). We acknowledge that the training corpus impacts the relations represented by word embeddings. Hence, the number of relevant diagrams identified may fluctuate based on the training corpus.

9. Conclusion

This paper presented SoCo, a semi-automated approach and a supporting tool to help organizations achieve software compliance with the GDPR DPPs. SoCo helps engineers identify data processing activities (modeled as sequence diagrams) that may be GDPR-relevant, so subject–matter experts can examine these for non-compliance using our tool. SoCo also supports engineers in selecting suitable privacy and security controls from a catalog and injecting them in sequence software

SECURITY CONTROLS' TEMPLATES DESCRIPTION

#1[Database Auditing]

For DPPs [7]
With new actors [AuditServer]
With sorting [AuditServer:1000]

A pointcut composed of [4] message(s) involving the [create, modify, view, erase] operations, the [$view, $service, $db] actors, and the [All] data types.
An advice of type [ADD AFTER] with the instructions:
$service -> AuditServer: appendAuditLog(userID, action)

#2 [Web Filtering]

For DPPs [6,7]
With new actors [WebFilter]
With sorting [WebFilter:0]

A pointcut composed of [4] message(s) involving the [request] operations, the [$view, $service] actors, and the [URL, Connection] data types.
An advice of type [REPLACE_ALL] with the instructions:
$view-> WebFilter: getConnection()
WebFilter-> $view: returnConnection()
$view-> WebFilter: requestURL(url)
WebFilter-> WebFilter: validateRequest(IP, url)
alt if url!=blackList
WebFilter -> $service: fwdRequest(url)
WebFilter <-- $service: response(url,status)
$view <-- WebFilter: response(url,status)
else else
$view <-- WebFilter: deny(url)
$view <-- WebFilter: closeConnection()
End

#3 [Valid Informed Consent]

For DPPs [1,7]
With new actors []
With sorting [Marketing:800]

A pointcut composed of [2] message(s) involving the [submit] operations, the [$view, *] actors, and the [personalData, email] data types.
An advice of type [WRAP AROUND] with the instructions:
$view<- Website: requestConsent()
alt if consentProvided
else else
$view <- Website: showError()
end

A pointcut composed of [1] message(s) involving the [submit] operations, the [*, Marketing] actors, and the [email] data types.
An advice of type [WRAP AROUND] with the instructions:
alt if optedInForMarketing
end

#10 [Logging]

For DPPs [4,7]
With new actors [Logger]
With sorting [Logger:900]

A pointcut composed of [1] message(s) involving the [modify] operations, the [*, $db] actors, and the [studentRecord] data types.
An advice of type [ADD AFTER] with the instructions:
$service -> Logger: writeLog(transaction)
Logger-> Logger: deidentifyLog(transactionDetails)
Logger-> Logger: writeTransactionLog(deIdentifiedTransaction)

#7 [ABAC]

For DPPs [2,3,6]
With new actors []
With sorting [AuthorizationEngine:400, AccessControlList:401]

A pointcut composed of [2] message(s) involving the [checkACL, checkPolicy] operations, the [AuthorizationEngine, AccessControlList] actors, and the [userID] data types.
An advice of type [REPLACE_ALL] with the instructions:
AuthorizationEngine-> AccessControlList: checkACL(asset, userID, userRole, purpose, userAtts, assetAtts)
AccessControlList-> AccessControlList: checkPolicy(user, action, resource, context)

A pointcut composed of [1] message(s) involving the [request] operations, the [$service, $db] actors, and the [studentRecord] data types.
An advice of type [ADD_AFTER] with the instructions:
AuthorizationEngine<- $db: checkAttsPermissions(asset, userID, userRole, purpose, userAtts, assetAtts)

A pointcut composed of [1] message(s) involving the [view] operations, the [$service, $db] actors, and the [studentRecord] data types.
An advice of type [REPLACE_ALL] with the instructions:
$service <-- $db: readFilteredStudentRecord

#8 [Retention Policy]

For DPPs [5]
With new actors [RetentionPolicyManager]
With sorting [RetentionPolicyManager:1200]

A pointcut composed of [2] message(s) involving the [store, save] operations, the [*, $db] actors, and the [CVOfCandidate] data types.
An advice of type [ADD BETWEEN] with the instructions:
$db->RetentionPolicyManager: obtainRetentionPeriod(CVOfCandidate)
RetentionPolicyManager->RetentionPolicyManager: getRetentionPeriod(CVOfCandidate)
$db<--RetentionPolicyManager: retentionPeriod
$db->$db: assignPolicyToDataItem(CVOfCandidate, retentionPeriod)

Fig. A.15. Security Control Templates (Part 1).

design diagrams to address GDPR violations. We conducted a case study with an IT professional who applied SoCo to a large software system and used the supporting tool in this evaluation. Our evaluation showed that SoCo could support engineers in various phases of the GDPR compliance exercise and facilitate the implementation of privacy and security controls in software systems. SoCo could successfully evolve complex sequence diagrams extracted from a large software system. Although relevant experience in security and privacy is desirable, our approach aims to compensate for this lack of expertise by selecting and applying appropriate controls in web software systems to adhere to GDPR DPPs (hence translating into significant time-savings for the engineers conducting the tasks).

CRediT authorship contribution statement

Vanessa Ayala-Rivera: Conceptualization, Data curation, Investigation, Methodology, Software, Validation, Writing – original draft, Writing – review & editing. **A. Omar Portillo-Dominguez:** Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Liliana Pasquale:** Conceptualization, Funding acquisition, Methodology, Project administration, Resources, Supervision, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

I have included the link to my supplementary material (which includes the code and data used) as reference in the paper.

Acknowledgments

This work was supported by Science Foundation Ireland (SFI) under Grants number 18/IF/6343, 10/CE/I1855, 13/RC/2094_2, and 15/SIRG/3501.

SECURITY CONTROLS' TEMPLATES DESCRIPTION

#4 [Encryption]

For DPPs [6]
With new actors [CryptoService]
With sorting [CryptoService:500]
A pointcut composed of [1] message(s) involving the [submit] operations, the [$service, *] actors, and the [sensitiveData] data types.
An advice of type [ADD_BEFORE] with the instructions:
$service -> CryptoService: selectEncryptionAlgorithm()
$service <-- CryptoService: encryptionAlgorithm
$service -> CryptoService: encrypt(message, algorithm)
$service <-- CryptoService: encryptedMessage

A pointcut composed of [1] message(s) involving the [view] operations, the [*, $service] actors, and the [sensitiveData] data types.
An advice of type [ADD_BEFORE] with the instructions:
$service-> CryptoService: decrypt(message, algorithm)
$service <-- CryptoService: decryptedMessage

#9 [Single Sign On]

For DPPs [6,7]
With new actors [IdentityProviderSSO]
With sorting [IdentityProviderSSO:1100]

A pointcut composed of [3] message(s) involving the [authenticate, verify] operations, the [*, $db] actors, and the [credentials] data types.
An advice of type [REPLACE_ALL] with the instructions:
$service-> IdentityProviderSSO: forwardForAuthentication(url)
$view <- IdentityProviderSSO: displaySingleSignOnPage()
$view -> IdentityProviderSSO: authenticateRequest(credentials)
IdentityProviderSSO-> IdentityProviderSSO: verifyCredentials(staffID)
$service <-- IdentityProviderSSO: authenticateResponseCredentials

#5[CookiesConsentBanner]

For DPPs [1,2,3]
With new actors []
With sorting []
A pointcut composed of [1] message(s) involving the [view] operations, the [$view, $service] actors, and the [ImpliedConsentCookie] data types.
An advice of type [REPLACE_ALL] with the instructions:
$view <-- $service: showOptInCookieConsentBanner(msg)

A pointcut composed of [2] message(s) involving the [accept, keep] operations, the [$view, $service] actors, and the [Cookies] data types.
An advice of type [REPLACE_BETWEEN] with the instructions:
else clickOnContinueBtn or Decline

A pointcut composed of [1] message(s) involving the [view, keep] operations, the [$view, $service] actors, and the [DefaultCookies] data types.
An advice of type [ADD_AFTER] with the instructions:
else clickOnManageCookieSettingBtn
$view -> $service: requestMoreInfoAboutCookies()
$view <- $service: showGramularCookieConsentBanner(cookieCategories)
loop
$view -> $view: selectCookies()
end
opt clickOnShowCookiesAndPrivacyPolicyBtn
$view <- $service: displayPrivacyStatement()
end

#6 [RBAC]

For DPPs [2,3,6]
With new actors [AuthorizationEngine, AccessControlList]
With sorting [AuthorizationEngine:400, AccessControlList:401]

A pointcut composed of [2] message(s) involving the [authorize, checkPrivileges] operations, the [$service, $db] actors, and the [userID] data types.
An advice of type [REPLACE_ALL] with the instructions:
$service-> AuthorizationEngine: checkAuthorization(studentID, userID)
AuthorizationEngine-> AccessControlList: checkACL(asset, userID)
AccessControlList-> AccessControlList: checkPolicy(user, role, permission, operation)
AuthorizationEngine <-- AccessControlList: userRolePrivileges

A pointcut composed of [1] message(s) involving the [access] operations, the [$service, $db] actors, and the [Granted] data types.
An advice of type [REPLACE_ALL] with the instructions:
$service <-- AuthorizationEngine: accessGranted

A pointcut composed of [1] message(s) involving the [access] operations, the [$service, $db] actors, and the [Denied] data types.
An advice of type [REPLACE_ALL] with the instructions:
$service <-- AuthorizationEngine: accessDenied

Fig. A.16. Security Control Templates (Part 2).

Appendix. Security controls templates

For further reference, Figs. A.15 and A.16 present the pre-filled templates for the 10 priv&sec controls used in this research work.

References

Aberkane, A.-J., Poels, G., Broucke, S.V., 2021. Exploring automated GDPR-compliance in requirements engineering: A systematic mapping study. IEEE Access 9, 66542–66559.

Acar, Y., Fahl, S., Mazurek, M.L., 2016. You are not your developer, either: A research agenda for usable security and privacy research beyond end users. (SecDev), 2016 IEEE Cybersecur. Dev. (SecDev).3–8,

Alhazmi, A., Arachchilage, N.A.G., 2021. I'm all ears! listening to software developers on putting GDPR principles into software development practice. Pers. Ubiquitous Comput. 25 (5), 879–892.

Ali, N., Jutla, D., Bodorik, P., 2015. PIP: An injection pattern for inserting privacy patterns and services in software. In: Annual Privacy Forum. Springer, pp. 144–157.

Alkubaisy, D., Piras, L., Al-Obeidallah, M.G., Cox, K., Mouratidis, H., 2021. A framework for privacy and security requirements analysis and conflict resolution for supporting GDPR compliance through privacy-by-design. In: ENASE. Springer, pp. 67–87.

Altova GmbH, Altova UModel, https://www.altova.com/manual/UModel/umodelbasic/umgenerate_code_from_sequence_di.html, (Accessed 27 February 2024).

Amaral, O., Abualhaija, S., Sabetzadeh, M., Briand, L., 2021. A model-based conceptualization of requirements for compliance checking of data processing against GDPR. In: 2021 IEEE 29th International Requirements Engineering Conference Workshops. REW, pp. 16–20. http://dx.doi.org/10.1109/REW53955.2021.00009.

Andrade, V.C., Gomes, R.D., Reinehr, S., Freitas, C.O.D.A., Malucelli, A., 2023. Privacy by design and software engineering: A systematic literature review. In: Proceedings of the XXI Brazilian Symposium on Software Quality. SBQS '22, Association for Computing Machinery, New York, NY, USA, ISBN: 9781450399999, http://dx.doi.org/10.1145/3571473.3571480.

Ayala-Rivera, V., Pasquale, L., 2018. The grace period has ended: An approach to operationalize gdpr requirements. In: 2018 IEEE 26th International Requirements Engineering Conference (RE).

Ayala-Rivera, V., Portillo-Dominguez, A., Pasquale, L., SoCo's GitHub Repository, https://github.com/oportillo78/SoCo, (Accessed 27 February 2024).

Ayala-Rivera, V., Portillo-Dominguez, A., Pasquale, L., SoCo's supplementary material and prototype tool, https://doi.org/10.5281/zenodo.11851250, (Accessed 17 June 2024).

Bodorik, P., Jutla, D., Bryn, A., 2014. Privacy engineering with PAWS: injecting restful privacy web services. IEEE Softw..

Boduch, A., 2016. Flux architecture. Packt Publishing Ltd.

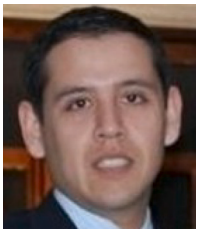
Bratthall, L., Jørgensen, M., 2002. Can you trust a single data source exploratory software engineering case study? Empir. Softw. Eng. 7 (1), 9–26.

Brazilian Federal Government, 2018. Brazilian general data protection law (LGPD). https://www.pnm.adv.br/wp-content/uploads/2018/08/Brazilian-General-Data-Protection-Law.pdf.

Bu, F., Wang, N., Jiang, B., Liang, H., 2020. "Privacy by design" implementation: Information system engineers' perspective. Int. J. Inf. Manage. 53, 102124.
Buse, R.P., Sadowski, C., Weimer, W., 2011. Benefits and barriers of user evaluation in software engineering research. In: Proceedings of the 2011 ACM International Conference on Object Oriented Programming Systems Languages and Applications. pp. 643–656.
Butler, S., Wermelinger, M., Yu, Y., 2015. Investigating naming convention adherence in java references. In: ICSME. IEEE.
Cachopo, J., Cruz, L., Fernandes, S., da Silva, F.M., Ribeiro, C., Rito-Silva, A., Ventura, A., 2011. The FenixEdu Project: an Open-Source Academic Information Platform. Instituto Superior Técnico.
California State Legislature, 2018. California consumer privacy act (CCPA). https://leginfo.legislature.ca.gov/faces/codes_displayText.xhtml?division=3.&part=4.&lawCode=CIV&title=1.81.5.
Cavoukian, A., 2012. Operationalizing privacy by design: A guide to implementing strong privacy practices. Ontario, Canada IPC.
Cavoukian, A., et al., 2009. Privacy by design: The 7 foundational principles, vol. 5, Ontario, Canada IPC.
Cejas, O.A., Abualhaija, S., Torre, D., Sabetzadeh, M., Briand, L., 2021. AI-enabled automation for completeness checking of privacy policies. IEEE Trans. Softw. Eng..
CIS, CIS PET wiki, https://cyberlaw.stanford.edu/wiki/index.php/PET, (Accessed 27 February 2024).
CIS, CIS Critical Security Controls, https://www.cisecurity.org/controls, (Accessed 27 February 2024).
Cohen, W.W., Ravikumar, P., Fienberg, S.E., et al., 2003. A comparison of string distance metrics for name-matching tasks. In: IIWeb, vol. 2003.
Colesky, M., Hoepman, J.-H., Hillen, C., 2016. A critical analysis of privacy design strategies. In: IEEE SPW. pp. 33–40.
Coleti, T.A., Corrêa, P.L.P., Filgueiras, L.V.L., Morandini, M., 2020. TR-model. a metadata profile application for personal data transparency. IEEE Access 8, 75184–75209.
Cortina, S., Picard, M., Renault, S., Valoggia, P., 2021. Towards a process-based approach to compliance with GDPR. In: EuroSPI. Springer, pp. 107–121.
Deloitte, Deloitte GDPR vision and approach, https://www2.deloitte.com/content/dam/Deloitte/nl/Documents/risk/deloitte-nl-risk-gdpr-vision-approach.pdf, (Accessed 27 February 2024).
Deng, M., Wuyts, K., Scandariato, R., Preneel, B., Joosen, W., 2011. A privacy threat analysis framework: Supporting the elicitation and fulfillment of privacy requirements. Requir. Eng. 16 (1), 3–32.
Dias Canedo, E., Toffano Seidel C., A., Toffano Seidel M., E., Teixeira Costa, P.H., Lima, F., 2020. Perceptions of ICT practitioners regarding software privacy. Entropy 22 (4), 429.
DPC Ireland, GDPR readiness checklist tools, http://gdprandyou.ie/wp-content/uploads/2017/12/A-Guide-to-help-SMEs-Prepare-for-the-GDPR.pdf, (Accessed 27 February 2024).
DPC Ireland, Data Protection Commission - Data Protection Impact Assessments, https://www.dataprotection.ie/en/organisations/know-your-obligations/data-protection-impact-assessments, (Accessed 27 February 2024).
DPC Ireland, 2019. Guidance Note: A Quick Guide To GDPR Breach Notifications. Report.
DPC Ireland, Data Protection Commission - Case Studies, https://www.dataprotection.ie/en/dpc-guidance/dpc-case-studies, (Accessed 27 February 2024).
Ehécatl Morales-Trujillo, M., García-Mireles, G.A., Matla-Cruz, E.O., Piattini, M., 2019. A systematic mapping study on privacy by design in software engineering. Clei Electr J.
Emeksiz, E., Zahadat, N., 2023. Ransomware as an imminent and destructive cyber-threat of the digital world. Available at SSRN 4644634.
European Parliament, Council of the European Union, 2016. 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). URL https://eur-lex.europa.eu/eli/reg/2016/679/oj.
Federal Trade Commission, 1996. Staff report.
Gali, N., Mariescu-Istodor, R., Hostettler, D., Fränti, P., 2019. Framework for syntactic string similarity measures. Expert Syst. Appl. 129, 169–185.
Guamán, D.S., Del Alamo, J.M., Caiza, J.C., 2021. GDPR compliance assessment for cross-border personal data transfers in android apps. IEEE Access 9, 15961–15982.
Hadar, I., Hasson, T., Ayalon, O., Toch, E., Birnhack, M., Sherman, S., Balissa, A., 2018. Privacy by designers: software developers' privacy mindset. Empir. Softw. Eng. 23 (1), 259–289.
Hernan, S., Lambert, S., Ostwald, T., Shostack, A., 2006. Threat modeling-uncover security design flaws using the stride approach. MSDN Magazine-Louisville 68–75.
Herzog, You Can't Have Privacy Without Security - National Cybersecurity Alliance. https://staysafeonline.org/cybersecurity-for-business/you-cant-have-privacy-without-security/.
Hoepman, J.-H., 2014. Privacy design strategies. In: IFIP IISC. Springer, pp. 446–459.
IAPP, 2022. Privacy Tech Vendor Report. Report.
ICO UK, How do we document our processing activities?, https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/consent/what-is-valid-consent/, (Accessed 27 February 2024).
ICO UK, UK GDPR guidance and resources, https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/, (Accessed 27 February 2024).
Instituto Superior Tecnico, FenixEdu, http://fenixedu.org, (Accessed 27 February 2024).
International Association of Privacy Professionals, Global Comprehensive Privacy Law Mapping Chart, https://iapp.org/resources/article/global-comprehensive-privacy-law-mapping-chart/, (Accessed 27 February 2024).
Isaca, GDPR Readiness, Assessment & Compliance, https://www.isaca.org/info/gdpr/index.html, (Accessed 27 February 2024).
ISO/IEC, ISO/IEC 27001, http://www.iso27001security.com/html/27001.html, (Accessed 27 February 2024).
ISO/IEC, ISO/IEC 27002:2022, https://www.iso.org/standard/75652.html, (Accessed 27 February 2024).
ISO/IEC, ISO/IEC 29100, https://www.iso.org/standard/45123.html, (Accessed 27 February 2024).
James, B., SDLC Risk Management Framework., https://www.oreilly.com/library/view/risk-management-framework/9781597499958/B9781597499958000053.xhtml, (Accessed 27 February 2024).
Jatnika, D., Bijaksana, M.A., Suryani, A.A., 2019. Word2vec model analysis for semantic similarities in english words. Procedia Comput. Sci. 157, 160–167.
Kabanov, I., 2016. Effective frameworks for delivering compliance with personal data privacy regulatory requirements. In: IEEE PST. pp. 551–554.
Kılıç, E., Sandıkkaya, M.T., 2023. Obfuscated JavaScript code detection using machine learning with AST-based syntactic and lexical analysis. In: 2023 8th International Conference on Smart and Sustainable Technologies. (SpliTech), IEEE, pp. 1–6.
Ko, A.J., LaToza, T.D., Burnett, M.M., 2015. A practical guide to controlled experiments of software engineering tools with human participants. Empir. Softw. Eng. 20, 110–141.
Köffel, C., John-Sören, P., Wolkerstorfer, P., Graf, C., Leif Erik, H., Ulrich, K., Hans, H., Benjamin, K., Stefanie, P., 2010. HCI pattern collection V2.
Krebs, FB Passwords in Plain Text, https://krebsonsecurity.com/2019/03/facebook-stored-hundreds-of-millions-of-user-passwords-in-plain-text-for-years/, (Accessed 27 February 2024).
KU Leuven, Linddun, https://linddun.org/, (Accessed 27 February 2024).
Kundu, D., Samanta, D., Mall, R., 2013. Automatic code generation from unified modelling language sequence diagrams. IET Softw. 7 (1), 12–28.
Lamprecht, A.-L., Garcia, L., Kuzak, M., Martinez, C., Arcila, R., Martin Del Pico, E., Dominguez Del Angel, V., Van De Sandt, S., Ison, J., Martinez, P.A., et al., 2020. Towards FAIR principles for research software. Data Sci. 3 (1), 37–59.
Lopez, T., Tun, T., Bandara, A., Mark, L., Nuseibeh, B., Sharp, H., 2019. An anatomy of security conversations in stack overflow. In: IEEE/ACM ICSE-SEIS. pp. 31–40.
Martin, Y.-S., Kung, A., 2018. Methods and tools for GDPR compliance through privacy and data protection engineering. In: IEEE EuroS&PW. pp. 108–111.
Maynooth University, Possible personal data in a university 2, https://www.maynoothuniversity.ie/data-protection/register-inventory-personal-data, (Accessed 27 February 2024).
Mcadam, T., RMS GDPR roadmap to compliance, https://www.rsm.global/ireland/sites/default/files/media/gdpr_roadmap_to_compliance_printable_version_-_terry_mcadam.pdf, (Accessed 27 February 2024).
Mouheb, D., Alhadidi, D., Nouh, M., Debbabi, M., Wang, L., Pourzandi, M., 2016. Aspect-oriented modeling framework for security hardening. Innov. Syst. Softw. Eng. 12 (1), 41–67.
Muller, C., Gurevych, I., 2009. A study on the semantic relatedness of query and document terms in information retrieval. EMNLP. pp. 1338–1347.
Nanayakkara, S., Perera, S., Weerasuriya, G.T., Gamage, P., Amarathunga, K., 2022. Software for IT project quality management. In: Managing Information Technology Projects: Building a Body of Knowledge in IT Project Management. World Scientific, pp. 411–450.
New Zealand Government, Learning from Privacy Incidents, https://www.digital.govt.nz/standards-and-guidance/privacy-security-and-risk/privacy/privacy-incidents-and-breaches/learning-from-privacy-incidents/, (Accessed 27 February 2024).
New Zealand Parliamentary Counsel Office, 2020. Privacy act 2020, New Zealand legislation. https://www.legislation.govt.nz/act/public/2020/0031/latest/whole.html.
NIST, 2020. Special publication 800-53 revision 4: Security and privacy controls for federal information systems and organizations. Tech. Rep., (800–53), National Institute of Standards and Technology (NIST), http://dx.doi.org/10.6028/NIST.SP.800-53r4.
ObjectAid, ObjectAid, http://www.edu4java.com/en/java-for-beginners/java-for-beginners16.html, (Accessed 27 February 2024).
OECD, The OECD Privacy Framework, https://www.oecd.org/sti/ieconomy/privacy-guidelines.htm, (Accessed 27 February 2024).
Paolone, G., Marinelli, M., Paesani, R., Di Felice, P., 2020. Automatic code generation of MVC web applications. Computers 9 (3), 56.
Parada, A.G., Siegert, E., De Brisolara, L.B., 2011. Generating java code from UML class and sequence diagrams. In: 2011 Brazilian Symposium on Computing System Engineering. IEEE, pp. 99–101.
Peixoto, M., Ferreira, D., Cavalcanti, M., Silva, C., Vilela, J., Araújo, J., Gorschek, T., 2020. On understanding how developers perceive and interpret privacy requirements research preview. In: REFSQ. pp. 116–123.

People's Republic of China, 2020. Personal information protection law of the People's Republic of China. https://personalinformationprotectionlaw.com/.
Pinheiro, J., 2020. Review of cyber threats on educational institutions. In: Proceedings of the Digital Privacy and Security Conference. Universidade Lusófona do Porto Porto, Portugal, p. 43.
Piras, L., Al-Obeidallah, M.G., Pavlidis, M., Mouratidis, H., Tsohou, A., Magkos, E., Praitano, A., 2021. A data scope management service to support privacy by design and gdpr compliance. J. Data Intell. 2 (2), 136–165.
Piras, L., Al-Obeidallah, M.G., Praitano, A., Tsohou, A., Mouratidis, H., Crespo, B.G.-N., Bernard, J.B., Fiorani, M., Magkos, E., Sanz, A.C., et al., 2019. Defend architecture: A privacy by design platform for gdpr compliance. In: TrustBus. Springer, pp. 78–93.
PlantUML, PlantUML, http://plantuml.com, (Accessed 27 February 2024).
Portillo-Dominguez, A.O., Perry, P., Magoni, D., Wang, M., Murphy, J., 2016. Trini: an adaptive load balancing strategy based on garbage collection for clustered java systems. Softw. - Pract. Exp. 46 (12), 1705–1733.
PRIPARE, Privacy Patterns, http://privacypatterns.eu/, (Accessed 27 February 2024).
Rauf, I., Lopez, T., Sharp, H., Petre, M., Tun, T., Levine, M., Towse, J., van der Linden, D., Rashid, A., Nuseibeh, B., 2022. Influences of developers' perspectives on their engagement with security in code. In: Proceedings of the 15th International Conference on Cooperative and Human Aspects of Software Engineering. pp. 86–95.
Rhahla, M., Allegue, S., Abdellatif, T., 2021. Guidelines for GDPR compliance in big data systems. JISA 61, 102896.
Robol, M., Breaux, T.D., Paja, E., Giorgini, P., 2022. Consent verification monitoring. arXiv preprint arXiv:2206.06406.
Runeson, P., Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. Empir. Softw. Eng. 14 (2), 131.
Runeson, P., Host, M., Rainer, A., Regnell, B., 2012. Case Study Research in Software Engineering: Guidelines and Examples. John Wiley & Sons.
SAS, SAS roadmap to GDPR, https://www.sas.com/en_gb/software/general-data-protection-regulation/sas-roadmap-to-gdpr.html, (Accessed 27 February 2024).
Senarath, A., Arachchilage, N.A., 2018. Why developers cannot embed privacy into software systems? An empirical investigation. In: EASE. pp. 211–216.
SonicWall, 2024 SonicWall Cyber Threat Report: Shifting Front Lines, https://www.sonicwall.com/threat-report/, (Accessed 27 February 2024).
Thevarmannil, M., 10 Types of Threat Modeling Methodology To Use in 2023, https://www.practical-devsecops.com/types-of-threat-modeling-methodology/, (Accessed 27 February 2024).
Torre, D., Alferez, M., Soltana, G., Sabetzadeh, M., Briand, L., 2020. Model driven engineering for data protection and privacy: Application and experience with GDPR. arXiv preprint arXiv:2007.12046.
Truong, N.B., Sun, K., Lee, G.M., Guo, Y., 2019. GDPR-compliant personal data management: A blockchain-based solution. IEEE Trans. Inf. Forensics Secur. 15, 1746–1761.
UCD, Possible personal data in a university 1, https://www.cs.ucd.ie/personal-data/, (Accessed 27 February 2024).
Ulven, J.B., Wangen, G., 2021. A systematic review of cybersecurity risks in higher education. Future Internet 13 (2), 39.
Vanezi, E., Kapitsaki, G.M., Kouzapas, D., Philippou, A., Papadopoulos, G.A., 2020. Diálogop-a language and a graphical tool for formally defining GDPR purposes. In: Research Challenges in Information Science: 14th International Conference, RCIS 2020, Limassol, Cyprus, September 23–25, 2020, Proceedings 14. Springer, pp. 569–575.
Visual Paradigm, Visual Paradigm The No.1 Development Tool Suite, https://www.visual-paradigm.com/, (Accessed 27 February 2024).
Votipka, D., Fulton, K.R., Parker, J., Hou, M., Mazurek, M.L., Hicks, M., 2020. Understanding security mistakes developers make: Qualitative analysis from build it, break it, fix it. In: 29th USENIX Security Symposium (USENIX Security 20). pp. 109–126.
Wuyts, K., Joosen, W., 2015. LINDDUN Privacy Threat Modeling: A Tutorial. CW Reports.
Yilmaz, S., Toklu, S., 2020. A deep learning analysis on question classification task using word2vec representations. Neural Comput. Appl. 1–20.

Vanessa Ayala-Rivera is a Lecturer at the School of Informatics and Cybersecurity within the Faculty of Computing, Digital, and Data at Technological University Dublin. She is also an Associate Faculty in the National College of Ireland. She received her Ph.D. from the School of Computer Science at University College Dublin in 2017. She is also an experienced software engineer in the IT industry. Her research interests include cybersecurity, secure software development, data protection and data privacy, privacy engineering, and regulatory compliance in software systems.



A. Omar Portillo-Dominguez is a Lecturer at the School of Enterprise Computing and Digital Transformation within the Faculty of Computing, Digital, and Data at Technological University Dublin. He received his Ph.D. degree from the School of Computer Science at University College Dublin in 2016. In his professional life, he has been a software development manager, researcher, and software engineer in industries such as telecom, cybersecurity, and fintech. His research interests include data security and privacy, requirements engineering, and software engineering.

Liliana Pasquale is a Lecturer and Programme Director of M.Sc. in Cybersecurity at University College Dublin, and a researcher at Lero, the Irish Software Research Centre. She received her Ph.D. degree from Politecnico di Milano in 2011. Her research interests include requirements engineering, security, privacy, and digital forensics. She has served in the Program and Organizing Committee of prestigious software engineering conferences, such as ICSE, FSE, ASE, RE. She is also part of the review committee of the IEEE TSE journal and the TOSEM journal.