Highlights

Assisting the Development of Privacy-aware Software:
The PRIAM Tooled Metamodel for GDPR

Selena Lamari, Nadjia Benblidia, Chouki Tibermacine, Christelle Urtado, Sylvain Vauttier

- A metamodel that formalizes the key concepts defined in the GDPR privacy regulation,
- A set of generic user stories capturing GDPR requirements,
- A domain-specific language and a database to manage privacy compliance,
- An MDE-based method to integrate privacy by design into existing applications,
- An expert-based evaluation of the metamodel, complemented by an assessment using LLMs.

Assisting the Development of Privacy-aware Software: The PRIAM Tooled Metamodel for GDPR

Selena Lamari [a,b], Nadjia Benblidia[b], Chouki Tibermacine[c], Christelle Urtado[d], Sylvain Vauttier[d]

[a] *LIRMM, Univ. Montpellier, CNRS, Montpellier, France*
[b] *LRDSI, Univ. Blida 1, Blida, Algeria*
[c] *IRISA, Univ. South Brittany, Vannes, France*
[d] *EuroMov Digital Health in Motion, Univ. Montpellier & IMT Mines Ales, Montpellier, France*

Abstract

Context: As software systems are more tailored to users, personal data is collected and exploited more than ever before. This situation raises the issue of user privacy protection. Conforming to personal data protection regulations, such as the European General Data Protection Regulation (GDPR), has thus become a legal obligation for application providers. However, there are no widely adopted proposals to formalize, implement, and assess compliance with the personal data privacy protection required by GDPR. **Objective:** In order to help developers, our overarching objective is to propose a tooled software engineering approach to add personal data protection capabilities to existing software, thus making software privacy-aware and GDPR-compliant. **Method:** We developed a method called PRIAM (PRIvacy Assessment Method) that goes beyond a conceptual description of the regulation by incorporating concrete, actionable software artifacts. This article presents the cornerstone of this method –PRIAM metamodel– along with its companion artifacts. **Results:** PRIAM metamodel captures the main concepts of GDPR and is then supported by a domain-specific language, user stories, and a dedicated database schema. The comprehensiveness and relevance of PRIAM metamodel have been qualitatively evaluated by GDPR experts through a questionnaire. Complementarily, an AI-based evaluation has been conducted, using some Large Language Models (LLMs), opening perspectives for fast, iterative evaluations of metamodels that formalize regulation texts. Besides, the practicality and usefulness of PRIAM metamodel and all its companion artifacts are highlighted through the running example of a *Sport center management application*, where privacy enforcement features, tailored to the specific personal data of the application, are generated and integrated. **Conclusion:** These two elements assert the viability of our proposal as a practical solution for assisting the development of privacy-aware applications that are compliant with GDPR requirements, thanks to customizable sets of actual development artifacts, systematically derived from a validated comprehensive formalization of the regulation articles.

Keywords: GDPR, Privacy Rights, Personal Data Protection, Privacy-aware Software, Model Driven Engineering, Metamodel, Domain-Specific Language, User Stories

1. Introduction

Today's software systems and services are tailored to user needs more than ever before, as they integrate user data collected during their usage and / or provided by IoT devices. This trend creates business opportunities but raises challenges in terms of user privacy protection. Personal data can be, from a technical perspective, easily collected, stored and transferred to other parties without asking for user consent and sometimes even without users being aware of it.
Various countries have enacted regulations aimed at protecting user privacy. The European General Data Protection Regulation (GDPR) is a well known example of such regulations. These regulations are long legal texts that are not easy to understand, master and put into practice. However, their enforcement has a concrete impact on software providers. For example, 2023 showcased various landmark cases for GDPR enforcement including a new record fine of 1.2 billion euros against a major web service company. As of December 2023, the total amount of all fines in European countries since May 2018 exceeds 4 billion euros[1]. As a consequence, privacy concerns now are a global challenge for businesses, as personal data protection regulations exist around the world.
This is amplified by the fact that privacy regulations often percolate outside their home countries: for example, all software, even when developed outside the EU, has to comply with the GDPR (GDPR, Art. 3) when sold to or used by EU citizens.

According to Ross Anderson [1]: "Privacy is the ability and / or right to protect your personal information and extends to the ability and / or right to prevent invasions of your personal space (the exact definition of which varies from one country to another). Privacy can extend to families but not to legal persons such as corporations.". This paper restricts the definition of privacy to that mediated by applications as applications can technically access and transfer personal information very easily. This paper also deals exclusively with personal data protection from an application point of view. For example, it excludes more technical issues that are often operating system, network or middleware-related such as securing data exchanges, user authentication or cryptology. Those issues are very important and constitute the necessary basis on which privacy concerns are to be dealt with at application level [1]. For the sake of simplicity, in the remainder of this paper we will mainly use the term *data privacy* when referring to *application-level personal data protection.*

Various research works explore data privacy in software development. For example, Palmirani *et al.* [2] define an ontology that models the main concepts of GDPR. Hoepman [3] proposes eight strategies (minimize, hide, separate, aggregate, inform, control, enforce and demonstrate) for designers to consider in order to enforce data privacy. These strategies can also be seen as GDPR requirements for the software development process. Various extensions of development processes have also been proposed that integrate best practices for data privacy enforcement [4, 5, 6]. Despite the extensive literature on the subject, to the best of

[1] https://www.enforcementtracker.com/?insights [Consulted 2024/06/10]

our knowledge, no practical solution currently exists to help developers seamlessly integrate data privacy concerns into their applications – for instance, enabling users to exercise their right to be forgotten or to easily access all stored personal data. Existing works either focus on a single privacy concern (*e.g.*, user consent [7]) or provide either declarative models or enhanced development processes that are not fully actionable in practice.

To tackle these identified issues, the overarching objective of our research is to propose a tooled software engineering approach to help developers add personal data protection capabilities to existing software. To do so, this paper extensively uses Model-Driven Engineering [8] and proposes PRIAM (PRIvacy Assessment Method) metamodel as its core artifact. PRIAM metamodel is built in a systematic manner from the GDPR. We captured the concepts described in this regulation text and specified them in PRIAM.
This metamodel serves as the foundation for various artifacts that collectively model the regulation and can be tailored for the specific business domains of software applications.
Accompanying artifacts comprise: i) a domain-specific language (DSL) derived from PRIAM metamodel, ii) generated user stories that embody GDPR requirements for a given software application and iii) a privacy enforcement database that stores application-specific as well as user-specific (non private, technical) data that are needed to implement users' rights. This proposal is original as it goes beyond a mere formalization of the regulation by offering concrete, actionable tools to support developers throughout the development process. Furthermore, PRIAM metamodel has been evaluated using two strategies: firstly qualitatively by surveying experts, who confirmed that the proposed metamodel meets GDPR requirements and, secondly, by conducting a complementary AI-based evaluation, using the latest LLMs.

The remainder of the paper is organized as follows. Section 2 provides background knowledge on GDPR. Section 3 sketches out our approach to provide developers with support tools to integrate privacy concerns into their software applications. Section 3 describes the *Sport center management application*, which serves as a running case study throughout the paper. Section 4 focuses on the presentation of PRIAM metamodel and explains how it covers GDPR principles. Section 5 exposes the implementation of PRIAM using MDE principles and tools with the JetBrains MPS metaprogramming platform, and explains how various artifacts were derived from it. Section 6 focuses on the evaluation of PRIAM metamodel and discusses the identified threats to validity and their mitigation. Section 7 presents related research works on privacy concerns. Section 8 draws conclusions and presents perspectives to this work.

2. Background Knowledge on Data Privacy Regulations and Focus on GDPR

Privacy protection has become central due to the rapid expansion of digital technologies and the massive accumulation of personal data.
Several regulations, such as GDPR in Europe, CCPA in California or PIPA in South Korea, illustrate the diversity of legislative frameworks that protect users' rights while taking into account economic and technological realities [9]. This paper focuses on GDPR because of its

global scope and comprehensive approach to personal data protection. Unlike CCPA (California Consumer Privacy Act), which limits the scope to a consumer who is a natural person residing in California, GDPR has a more holistic approach. It imposes strict obligations on organizations and emphasizes particularly individual rights of people. GDPR applies to European Union (EU) residents, regardless of whether or not they actually are inside the EU, thus giving it an international dimension. Other regulations, such as South Korea's PIPA (Personal Information Protection Act), tend to of harmonize and become more in line with global standards for personal data protection, as defined by the GDPR.

GDPR emphasizes fundamental principles governing personal data processing. These include:

- *Lawfulness*: personal data processing must be based on one of the legal bases listed in the regulation. There are several legal bases for processing data: consent of the data subject, performance of a contract, compliance with a legal obligation, protection of vital interests of the data subject, performance of a task of public interest or legitimate interest pursued by the data controller.

- *Fairness*: personal data must not be misused or used in a way that could harm the data subject.

- *Transparency*: data subjects must be fully informed about the processing activities related to their personal data, their purposes and the categories of personal data being processed. Information must be provided transparently, in a clear and accessible language.

- *Purpose Limitation*: personal data must be collected for clear, explicit and legitimate purposes. Processing must not go beyond these initial purposes and may not be extended to other purposes incompatible with those for which the data was collected.

- *Data Minimization*: the application owner should only collect and process data that is adequate, relevant and necessary for the purposes of processing.

- *Accuracy*: personal data must be accurate and reflect reality. If errors or inaccuracies are found, they must be corrected without delay.

- *Storage Limitation*: personal data must not be retained for longer than necessary to achieve the purposes for which it was collected. Once this period has expired, data must be deleted or anonymized, so that it can no longer be used to identify the related data subject.

- *Integrity and Confidentiality (Security)*: personal data must be processed in such a way as to guarantee its security, integrity and confidentiality. This includes implementing technical and organizational measures to prevent unauthorized access, accidental disclosure, modification or destruction and loss of data.

- *Accountability:* the data processor holds responsibility for GDPR compliance and has to be able to demonstrate it. This involves implementing prescribed measures, such as keeping records of processing activities (GDPR, Art. 30).

Other concepts and requirements also deserve attention, particularly with regard to the elements needed to implement the regulation.

- *Data subjects rights:* GDPR strongly emphasizes individual rights, enabling data subjects to have control over their personal data. They can benefit from several rights: access, rectification, deletion, limitation of processing, data portability and objection (GDPR, Chap. 3).

- *Consent:* data subjects must be able to give consent to their data processing in a free, specific, informed and unambiguous manner. They must also be able to withdraw their consent as easily as they gave it.

- *Child's consent:* when a data subject is under 16 (minor), it is up to the holder of parental responsibility to give their consent. However, the age threshold for a data subject to provide their own consent varies from an EU member state to the other, ranging from 13 to 16 (GDPR, Art. 8).

- *Transfer outside the EU:* GDPR provides a framework for the transfer of personal data to countries outside of EU. This is only permitted if the recipient country offers an adequate level of protection for personal data or if appropriate safeguards, such as standard contractual clauses, are set to ensure data security (GDPR, Art. 44-50).

- *Breaches management:* Any security breach that compromises confidentiality, integrity, or availability of personal data is considered to be a personal data breach. When such a breach occurs, its impacts have to be analyzed in order to notify authorities and / or impacted data subjects. If the breach is likely to result in a risk to the rights and freedoms of data subjects, it must be notified to the supervisory authority within 72 hours. In the event of a high risk, data subjects must also be informed as soon as possible (GDPR, Art. 33,34).

After this overview of GDPR, the following section describes the proposed approach to integrate its requirements into software systems.

3. Privacy Integration Approach

This section first presents the proposed privacy integration approach in a nutshell, before providing details in subsequent sections. Second, it details the different user roles involved in the proposed approach. Finally, it describes a running example of a `Sport center management` application.

3.1. Privacy Integration Approach: Stages and Artifacts

The overarching ambition of our research is to assist developers in making their software privacy-aware as defined by GDPR. Figure 1 depicts the global approach to model privacy concerns and integrate them into applications in a tooled manner. This approach decomposes into three stages. The first two are inspired from software product line engineering [10]: *I. Privacy Rights Domain Engineering* involves modeling privacy enforcement requirements derived from regulations and implementing them as reusable and customizable artifacts; *II. Privacy-aware Application Engineering* focuses on tailoring a specific application by adapting and integrating the reusable artifacts from the first stage to ensure privacy compliance within that application. A third and final stage corresponds to the operation of the privacy-aware application at runtime (*III. Privacy-aware Application Operation*). These three stages are described below.

Stage I. Privacy Rights Domain Engineering. The first stage of our approach consists in formalizing the regulation as a metamodel. This metamodel is then used as the core of an MDE (*Model-Driven Engineering*) framework composed of:

- A Domain-Specific Language (DSL), whose syntax is derived from the metamodel, provides a mechanism in the next stage to declare personal data and their processings.
- a set of 65 generic user-stories that result from both our analysis of the regulation and a comparison with state of the art research [11]. These generic user-stories are used at the next stage to automatically generate application-specific user-stories from the annotations written with the DSL.
- a generic privacy enforcement database schema. This schema is used at the next stage to store: i) application-specific information needed for privacy-right enforcement, such as lists of collected and processed personal data, and ii) user-specific information, such as requests for excercising a rectification right.
- a set of microservices implementing the user stories and enabling to manage the aforementioned application-specific and user-specific information.

The first three artifacts are the output of the first step of this stage: *I.a. Privacy rights modeling*. The last artifact is the result of the second step: *Privacy rights implementation*. This step is not detailed in this paper.

Stage II. Privacy-aware Application Engineering. The second stage focuses on integrating privacy enforcement into applications by leveraging the DSL together with the previously introduced generic artifacts. The first step in this stage, called *Privacy Requirements Elicitation*, produces application-specific artifacts, including:

- an application-specific instance of the metamodel. This instance is obtained by using the DSL to declare personal data and their processings.

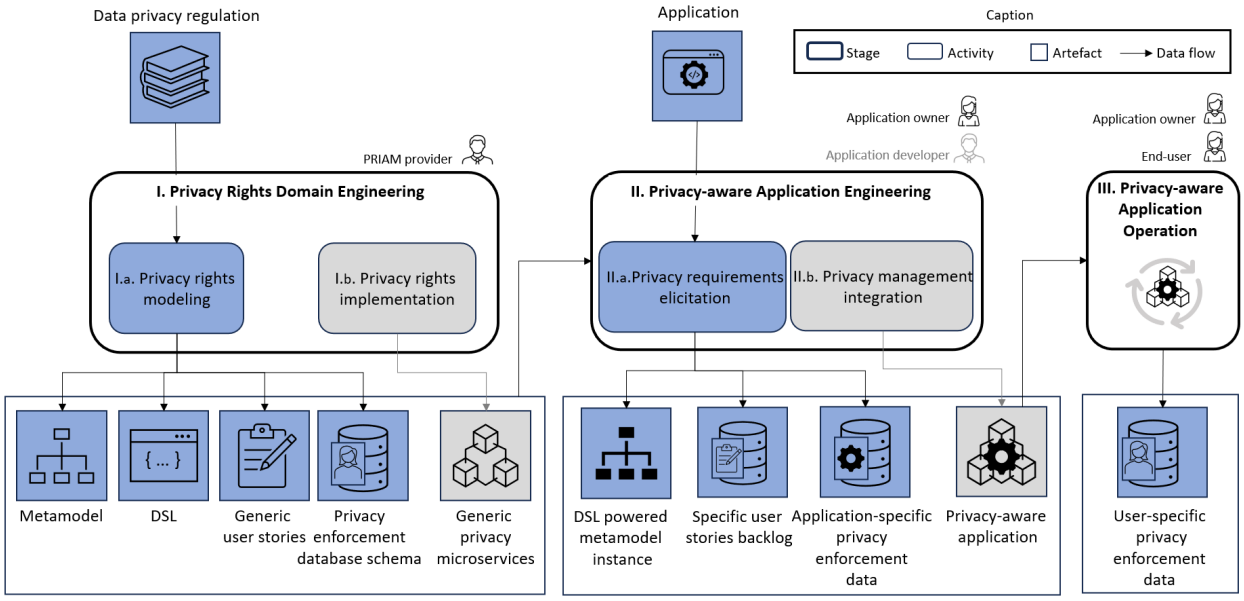

Figure 1: The proposed privacy integration approach stages and produced artifacts

- an application-specific list of user stories that are ready to be inserted in the application backlog to trace their implementation into the application.
- a privacy enforcement database populated with application-specific privacy enforcement data such as the list of processings that will claim consents to use personal data or data categories that are considered personal. This application-specific data is derived from declarations that are set via the DSL.

The second step of this stage involves integrating the privacy enforcement microservices into the application. This includes, among other tasks, the use of standard protocols for user authentication and authorization, with the aim of being as non-intrusive as possible within the application. These aspects fall outside the scope of this paper and will therefore not be discussed further. We make here the assumption that after this step the existing application is enriched with privacy enforcement features (right to access and rectify personal data, for example). We call the resulting application *privacy-aware application*.

Stage III. Privacy-aware Application Operation.
The third stage of the approach corresponds to the lifetime of the application where end-users and application owners interact with privacy enforcement features at runtime. The artifact produced at this third stage is user-specific and consists in the population of the privacy enforcement database with user-specific data. This includes end-user consents and (access, rectification, ...) rights-related requests in addition to application owner answers to these requests.

In Figure 1, all steps and artifacts that are described in this article are colored in blue. They

constitute the parts of the framework that are directly derived from the metamodel, which is the main contribution of our work here. Grey-colored items are not described here, as they fall outside the scope of this paper. Further details about them can be found in a previous publication [12].

3.2. Privacy Integration Approach: Roles

Our Privacy Integration Approach involves four distinct roles that altogether contribute to implement privacy-aware applications:

- **PRIAM provider**. PRIAM provider is responsible for Privacy Rights Domain Engineering (*cf.* first stage in Figure 1). They model privacy rights from the Data Privacy Regulation (GDPR). It results in a metamodel and several generic accompanying artifacts that will be used further to make applications privacy-aware. This role is fulfilled by the authors of this paper. As it is a generic, application-independent, engineering stage, PRIAM framework is provided once for all and is used for each specific application.

- **Application Owner**. The Application Owner is responsible for Privacy requirements elicitation in the Privacy-aware Application Engineering stage (*cf.* second stage in Figure 1). The Application Owner is an expert of the application and its data. They declare data types to make sure that every personal data in the application is dealt with as required by GDPR. This privacy requirements elicitation is a necessary step for application-specific privacy concerns to be further taken into account systematically and automatically. The Application Owner is referred to as the Data controller in GDPR. They are a legal person embodied by their legal representative (GDPR, Art. 4). They define the purposes and means of each processing that involves personal data. Their identity and contact details must be provided to end-users (GDPR, Art. 13.1). During Privacy-aware Application Operation, the Application Owner also answers end-user requests to exercise their rights, and provides, if necessary, information on identified data breaches.

- **Application Developer**. The Application Developer is an IT person that masters the domain of the application. They are the person responsible for integrating privacy concerns into the application.

- **End-user**. The End-user is the person that uses the application. Their personal data is collected and are the target of the privacy-preservation mechanisms as described in GDPR. The End-user is also referred to as Data Subject in GDPR. They have the right to control access to their personal data in order to protect their privacy.

3.3. Privacy Integration Approach: Privacy-Aware Application Engineering Activities

To better understand how our approach integrates privacy concerns into applications, we now present the activities of stage two of the privacy integration process. This stage decomposes into two activities.

1. Privacy requirements elicitation. At this step, the application owner is responsible for annotating personal data and their processings. They leverage the DSL in this annotation task. This step consists in identifying: (i) actors (*e.g.*, application owner and end-users), (ii) (personal) data, (iii) processings, including their links to personal data and their purposes.

Annotations expressed with the DSL lead to the automatic generation of: (i) *specific user stories* that are gathered into the application backlog, and (ii) *a database for privacy enforcement*, populated with application-specific privacy enforcement metadata.

2. Privacy management integration. In this activity, the application developer must update their application so that it takes all GDPR requirements into account. This consists in implementing all the application-specific privacy user stories generated during the privacy requirements elicitation activity.

To perform this task, the application developer is provided with the operational set of generic microservices implementing GDPR requirements resulting from the privacy rights implementation activity[12]. The microservices leverage the application-specific privacy enforcement metadata previously generated. that contains application-specific privacy enforcement data. The result of this step is a privacy-aware application.

3.4. Privacy Integration Approach: Privacy-aware Application Operation

Once the privacy-aware application has undergone deployment, it stands poised for operational functionality. As a GDPR-compliant system, it enables the seamless management of user consent, facilitates the exercise of user rights related to data rectification and erasure, objection to data collection / transfers, or data processing limitations. Moreover, it diligently records processing activities and automates breach response mechanisms wherever feasible. Throughout its operational lifespan, the application diligently collects privacy management data from its end-users, ensuring ongoing compliance and user protection.

Although the proposed solution enables customization of privacy management for given applications, it more generically standardizes privacy management, in the sense of GDPR. The proposed approach enables application engineers to integrate "standard" services for managing privacy (*e.g.*, consents and rights). Customization (by DSL annotation) in this solution is limited to application-specific "personal data collection" and "processing activities". The way privacy is enforced is the same in whatever application and/or industry. This enables to reach a good trade-off between customization and standardization.

3.5. Case Example

A case example is used throughout the paper to illustrate our work. It is an application operated by a sports center, which serves as a storehouse for users' personal data. This includes identification details: first name, email, gender, age, password, etc. The application also manages data specific to different user categories, such as skills for trainers and members' physical profiles (*see* Figure 2). Members use Internet of Things (IoT) devices

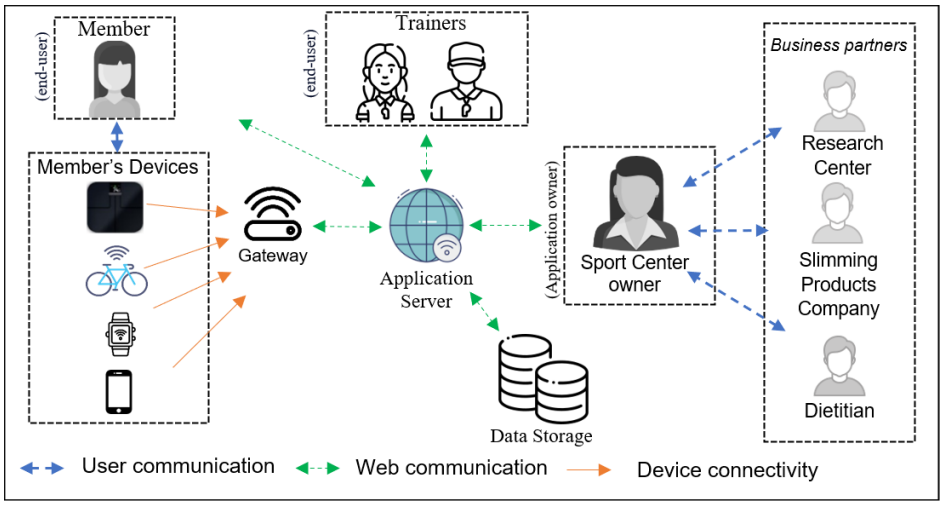

Figure 2: Sport center management application.

to monitor their activities to allow for the customization of training programs by trainers. Moreover, the sports center manager, who plays the role of application owner, collaborates with other partners for data sharing. These partners include a dietitian, a research center, and a company selling slimming products.

In order to comply with GDPR principles [13], end-users (members and trainers) should first be informed about all the categories of personal data collected by the application and about all their usages (processing or transmission to third parties). Second, end-users should explicitly express their consent for every collection or usage of personal data. Third, they should be able to revise their consents. Finally, end-users should be able to consult the collected data and ask for rectification, deletion or export (for their portability). This way, end-users fully control their personal data.

The application described above was developed with a focus on its functional requirements and some additional considerations such as maintainability, security, and reliability. However, during the development process, privacy –a crucial extra-functional requirement– was not adequately addressed, resulting in the application not being GDPR-compliant. For instance, the data collected from the connected objects and stored in the application server is exploited without any notice to or control from the end-users.

Figure 3 shows an excerpt of the Sport Center application's relational schema[2]. It consists of five tables. `member`, represents the sports center's members, that can participate in training sessions with trainers through the `trainer_booking` association. Each `trainer` is specialized

[2]Both the relational schema and the record example are extracted from a database proposed by `https://github.com/redhawk96/Fire_Fitness/blob/master/project1.sql` which we used to elaborate our example application.

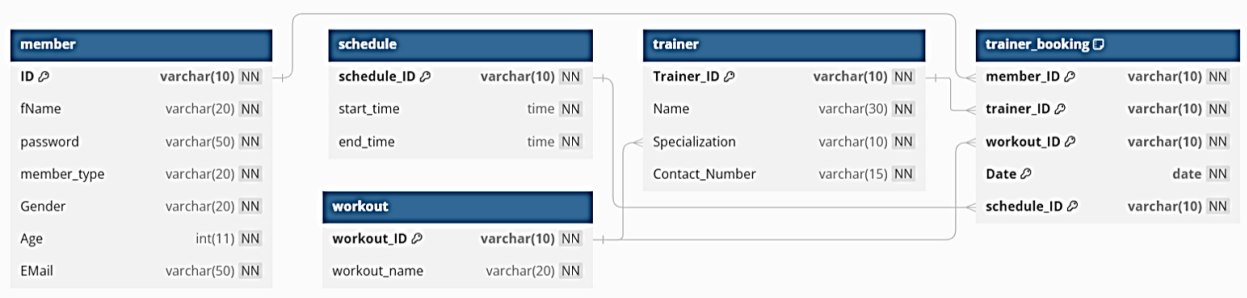

Figure 3: Excerpt of the sport center database schema.

in a type of training, represented by the `workout` table. Training sessions are planned based on a `schedule`, which gathers available training slots. Figure 4 provides a closer view of a single record from the `member` table. These tables are used as an example of personal data in the paper.

member

ID	fName	password	member_type	Gender	Age	EMail
M002	Sohan	sohan96	Silver	Male	21	sohan96@gmail.com

Figure 4: Illustrative example of a `member` table record.

4. PRIAM Metamodel

This section describes the PRIAM metamodel that captures the concepts of GDPR. This metamodel is decomposed in a set of packages, each dedicated to a key concept of the GDPR. The metamodel is enriched by a set of well-formedness rules, written using the OCL constraint language, that precises and narrows the metamodel instantiation possibilities.

4.1. Metamodel Packages Description

In order to build a metamodel that gathers GDPR concepts, we undertook a meticulous process to understand GDPR. The first step was an initial reading of the full text of the GDPR to gain an overview of the regulation. The second step was to read it again and deepen the analyzes to extract key requirements and fundamental concepts. We started the design of the PRIAM metamodel as a single large model of more than 45 classes. Then, we clustered these classes into seven main packages, corresponding to seven subdomains: `Actors`, `Data`, `Processings`, `Consents`, `Rights`, `Breaches` and `Documents`. They are depicted in Figures 5 to 13 and are presented in the following paragraphs.

4.1.1. Actors package.

The `Actors` package (*see* Figure 5) models the different kinds of stakeholders of the application. Everyone has a specific role. Two types of actors can be defined. `Main Actors` correspond to entities playing central roles in the privacy management process, while `Secondary`

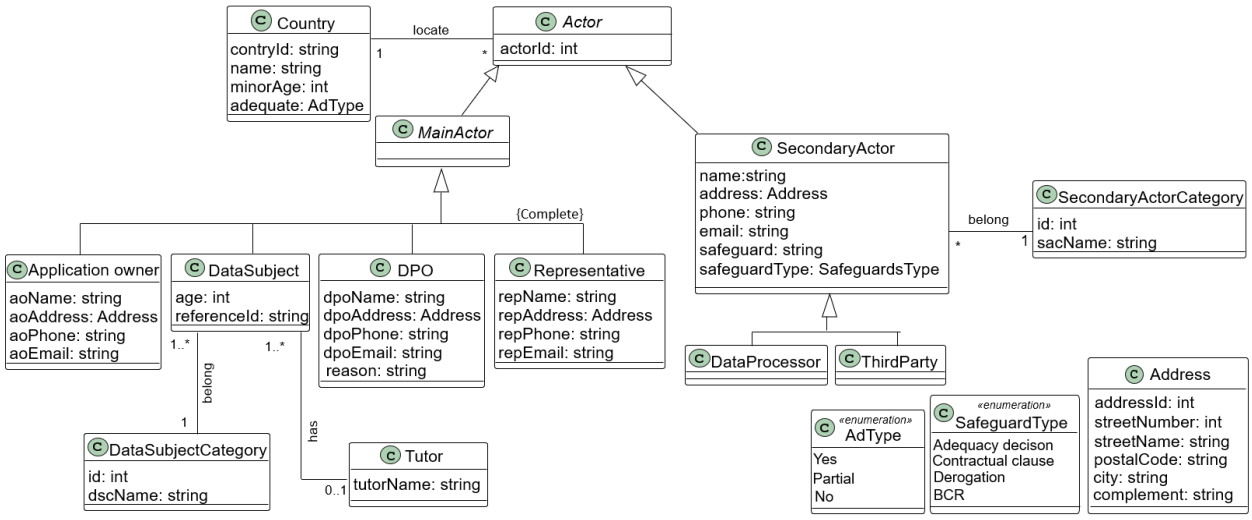

Figure 5: Actors package.

`actors` corresponds to external entities to which personal data is transmitted. Six types of actors are identified:

- The `application owner` and the `data subject` are defined in Section 3. In the example of Figure 2, the `Sport center owner` is considered the `application owner` and `members` and `trainers` to be `data subjects`.
- A `DPO` (Data Protection Officer) is responsible for enforcing personal data regulations within an organization (*e.g.*, company). A `DPO` is recommended in general and mandatory in several identified cases, such as when the organization uses particular data categories on a large scale (GDPR, Art. 37). `DPOs` check compliance with GDPR rules, inform and advise the organization, and also act as an interface between the organization, the supervisory authority (one of the predefined `SecondaryActorCategory`) and data subjects. As such, their contact details (*e.g.*, `name`, `address`, `email`) must be recorded (GDPR, Art. 37.1).
- A `SecondaryActor` is a natural or legal person entitled to obtain communication of the collected or produced data. They can be third-parties[3] or a processor. Every secondary actor belongs to a category (*e.g.*, commercial partner, research center, or supervisory authority). In the example of Figure 2, `Research Center`, `Slimming Products Company` and `Dietitian` are secondary actors. Secondary actors do not have access to the application (as they are external to the organization) but they can nonetheless manage data breaches (*see* Breaches in page 19) by providing breach-related information.

[3]"Third party means a natural or legal person, public authority, agency or body other than the data subject, controller, processor and persons who, under the direct authority of the controller or processor, are authorized to process personal data." (GDPR, Art. 4).

- A `Tutor` is a person holding parental responsibility[4] on a data subject who is considered minor (GDPR, Art. 8). In that case, consent cannot be provided by the data subject and has to be given by their tutor. The age for minority is set to 16 years, by default, but can be customized by European Union (EU) member states in the range [13 - 16]. In order to manage this customization, a member state minority age is recorded using the `minorAge` attribute of the `Country` class.

- A `Representative` is a natural or legal person established in the EU who represents the data controller or processor[5] of an organization located outside the EU. `Representatives` act as a communication point between the supervisory authorities and the data controller or processor. As such, their contact details (*e.g.*, `name`, `address`, `email`) must be recorded.

Each actor has an identifier. A data subject has a reference ID. This attribute records the information used to identify the data subject in the application databases. It is used as a means to link the metadata managed by the privacy enforcement database to the data managed by the application databases. For example, in the database record of Figure 4, `Sohan` is a data subject whose reference ID is M002.

Each actor is associated with a `Country`, which determines their geographical location. This is useful, for example, to assess whether secondary actors are established outside the EU. Establishment countries are needed to comply with regulations relating to cross-border / international data transfers. If a secondary actor is from outside the EU, we check whether its data protection regulation fully covers, partially covers or does not cover GDPR rules[6]. The `adequate` attribute of the `Country` class is used for this purpose. If its `adequate` value is `no`, the `secondary actor` must provide safeguards which are then recorded as a `String` in its `safeguard` attribute.

4.1.2. Data package.
The `Data` subdomain models all application data and documents, personal or not. `Data` holds the description of a kind of collected and stored data. Personal data (`Data` instances for which attribute `isPersonal` is set to `true`) have a non `null` `DataConservationDuration` attribute as personal information cannot be kept for an indefinite period of time (GDPR, Art. 13.2a, 14.2a, 15.1d). The `Source` of personal data (*i.e.*, whether the data subject directly provides data or not) (GDPR, Art. 14.2f, 15.1g) can be `Direct`, `Indirect` and obey distinct rules.

[4] In the context of GDPR.

[5] "Processor means a natural or legal person, public authority, agency or other body which processes personal data on behalf of the controller" (GDPR, Art. 4).

[6] For example, the french National Computer Science and Liberties Commission (CNIL) provides a list of those compatibility levels on its website: `https://www.cnil.fr/fr/node/163811`

`DataType` defines the structure of a type of data (in the sense of a classifier). In the example of Figure 3, each table corresponds to a `Datatype`: `workout`, `schedule`, `member`, `trainer` and `trainer_booking`. Each instance of `Data` documents a structural feature of a `DataType`, stating for instance if a piece of data is personal (*e.g.*, `name` in the `trainer` table) or non-personal (*e.g.*, `workout_name` in the `workout` table). `name`, as a personal data, is assigned to both a `PersonalDataCategory` (`identification data`) and a `DataSubjectCategory` (`trainer`). Its `source` is set as `Direct` as it is entered by the trainer when they register in the application. It therefore does not require any `SourceDetails` (attribute value is set to `null`).

The relationship between `DataSubjectCategory` and `Data` (*e.g.*, between `member` and `Data` with the `dataName fname` in Figure 3) helps filter application `Data` related to a specific kind of `DataSubject`. Figure 7 presents a complete example of an object diagram illustrating this relationship.

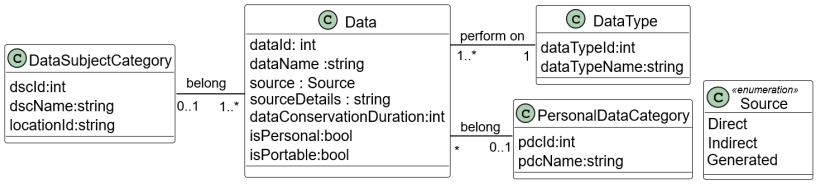

Figure 6: Data package.

4.1.3. Processings package.
The Processings package (*see* Figure 8) describes the functionalities of the application and the interactions carried out by actors. A `Processing` is an operation or set of operations performed on personal data (GDPR, Art. 4). Processings are of different types. `Default` processings do not require the consent of data subjects. For example, the signing-in of a data subject is a default processing. `Necessary` processings are the basic functionalities of an application. The consent of the data subject must be granted in order to use the application (*e.g.*, follow-up of the evolution of the member's weight). `Mandatory` processings are lawful processings that are not contractual ones or those requiring consent (*e.g.*, legal processings). `Optional Processings` do not alter the basic operation of the application when consent is refused or withdrawn by users (*e.g.*, profiling).

A processing on personal data must belong to a well-defined `ProcessingCategory` (GDPR, Art. 6) such as `Consent`. Besides, a processing is executed to achieve at least one `Purpose` (*i.e.*, functional objective of the application). All purposes of a processing must clearly be communicated to data subjects (GDPR, Art. 12).

`DataUsage` links processings to the data they use to assess their status regarding privacy management. Keeping track of the data used for each processing operation, as well as their

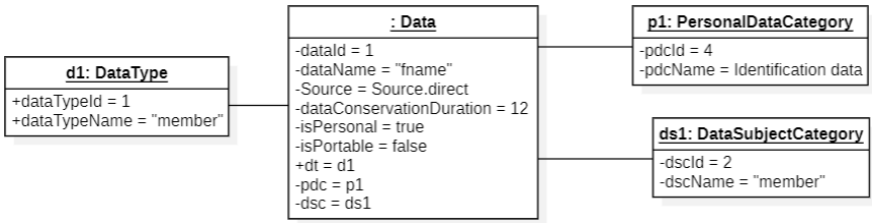

Figure 7: Object diagram for the Data package.

access rights to that data, contributes to satisfying the principle of minimization, providing a means of verifying that the personal data collected, processed and stored is reduced to what is strictly necessary to achieve the processing's purposes. In the context of a processing, personal data can be used without necessarily identifying the `DataSubject`, for example, calculating the average age of members. In this case, the age would be given a `personalStatus` attribute set to `false`, as the processing does not compromise users' privacy. This capability offers the application owner the possibility to manipulate personal data in a non personal way, thus providing extra flexibility without compromising GDPR rules.

A `ProcessingLink` links two processings together when necessary. There are two types of links: `Similarity` when processings have the same purposes and `Inclusion` when one processing includes the other. In both cases, a single consent is sufficient (GDPR, Art. 6.4) which simplifies consent management.

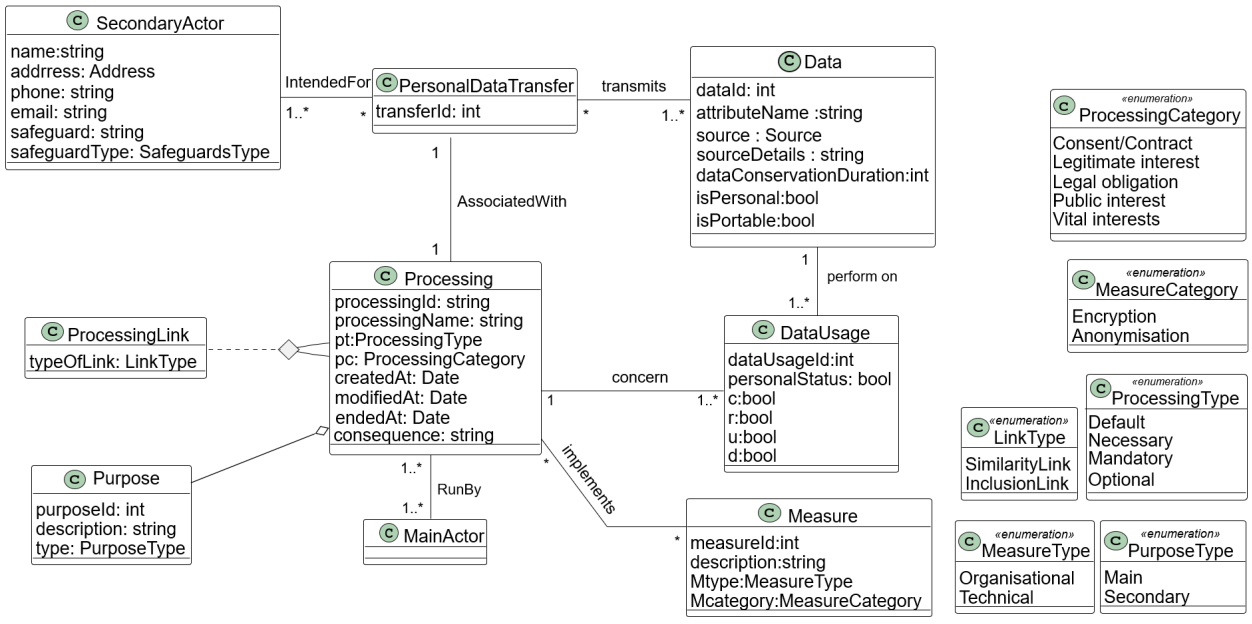

Figure 8: Processings package.

The `PersonalDataTransfer` class keeps track of any data transfer. It allows to know to

Processing	
processing Name	The transmission of data for targeted advertising purposes
pt	Optional
pc	Consent/Contract
createdAt	12/04/2022
modifiedAt	null
endedAt	null

Purpose 1	
description	Use data to personalize promotional offers and product recommendations according to members' preferences and training habits
type	Main

Purpose 2	
description	Analyze member preferences to improve sales and enhance customer satisfaction.
type	Secondary

Secondary actor	Used **Data**
Slimming products company	fname, gender, age, email, workout_name

Figure 9: A processing description example.

whom the data has been transferred to (via its association to `SecondaryActor`) and for what purpose (via its associated `Processings`' `Purposes`). GDPR strongly emphasizes the role of common processings, such as profiling or automated decision making, that massively involve personal data. Those are typical examples of possible instances of the `Processing` class. The `Measure` class serves as a log for whatever measures are implemented to mitigate data security risks.
Figure 9 provides a concrete example of a processing that can be carried out within the sports center[7]. It is related to targeted advertising purposes. To do so, the sports center transmits personal data to a `Slimming products company`, which uses this information to send personalized advertising to sport center members. This processing should be described as shown on Figure 9.

The processing is `optional` and therefore requires the consent of the data subject (the sport center's `Member`) before it can be carried out by the `Sport Center owner`. The processing requires the `Member`'s personal data, in particular their `email` address, to send targeted advertising directly to their email inbox, and `workout_name`, which provides information about their fitness and wellness interests. All used data is read-only (in `DataUsage`, r is set to `true`).

4.1.4. Consents package.
The `Consent` package manages and logs `DataSubjects`' `Consents` to operate given `Processings` (*see* Figure 10). It records details about `Consents` (*e.g.*, dates of their granting and possible withdrawal), logs the history of their modifications.

In order to guarantee the lawfulness of the application, `DataSubjects` should be informed beforehand about all `Processings` that plan to use their personal data (GDPR, Art. 13). It is a good practice to do so before using the application, during the registration phase, so that

[7]Instead of illustrating this example as an object diagram, for reasons of readability, we present a tabular (more concise) description.

DataSubjects can decide to either restrict their use of the application or, more radically, uninstall the application before it starts collecting and processing any of their personal data. If a DataSubject agrees to use the application, a Contract is assigned to them. A Contract groups together all Consents related to a DataSubject. DataSubjects must explicitly give their consent for each processing, when knowing precisely what personal data is required (GDPR, Art. 6). If the data subject does not consent, the processing should be blocked. For example, if the data subject refuses automatic profiling, access to their personal data by this processing should not be possible. Our proposed approach enables to enforce this rule.

As described briefly in the Actors package, GDPR provides specific Consent management rules for minors (GDPR, Art.8). The age attribute of the DataSubject class is recorded for this purpose and compared to its Country's minorAge value in order to decide if Consent is to be provided by DataSubject themselves or by their Tutor.

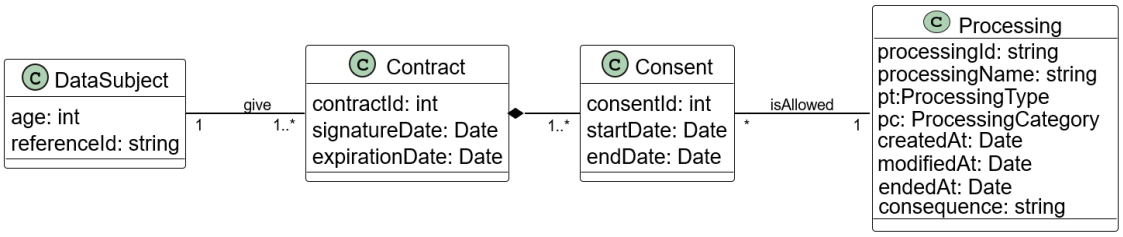

Figure 10: Consents package.

To illustrate how consents are managed, let us imagine Sohan, one of the sport center's Members (*see* Figure 2) wishes to consent to the processing Transmission of data for targeted advertising purposes as described in Figure 9. To do so, Sohan, as the DataSubject, gives his explicit consent through the application by filling in a form where he checks consents to the processing by ticking a checkbox. The Transmission of data for targeted advertising purposes processing is then added to the list of processings Sohan has given his consent to, recording Consent's startDate (date at which consent is given). Sohan may withdraw this consent at any time, and an endDate for the consent would equally be recorded.

4.1.5. Rights package.

The Right package describes the contractual relationship between the DataSubject and the ApplicationOwner relatively to the DataSubject's private data conservation and usage by Processings (GDPR, Art. 16-21). A DataSubject has the right to Access, Rectify or Erase their personal data as well as ask for their personal data Portability.

A DataSubject also has the Right to know what Processings are using their personal data for (Knowlegde), to object to Processings using their personal data (Objection) or to restrict the use of their personal data, like in the case where personal data must not be used by the application but still be kept for the data subject defense in a trial (Restriction).

These `Rights` can be exercized through `Requests` by a `DataSubject` to the `ApplicationOwner`. These can be of two types (subclasses of the `Request` abstract class): `DataRequests` for those requests related to data conservation and `ProcessingRequests` for those related to personal data usage by `Processings`. Each `Request` must be answered (links between `Request` and `Answer` instances are created).

`Requests` and their `Answers` are saved together with their `issuedAt` date and `claim` description. The date of a `Request` can be used to alert the `ApplicationOwner` if they have not answered the request in a timely manner[8].

If a `DataSubject` issues a `Rectification` request, they must provide the new desired value (`newValue` attribute of the `DataRequest`). `Rectification` contributes to the GDPR fundamental principle of *Accuracy* (as discussed in Section 2) because the `DataSubject` can check their data and ask for changes. To comply with GDPR Art. 18.1a, we provide the possibility

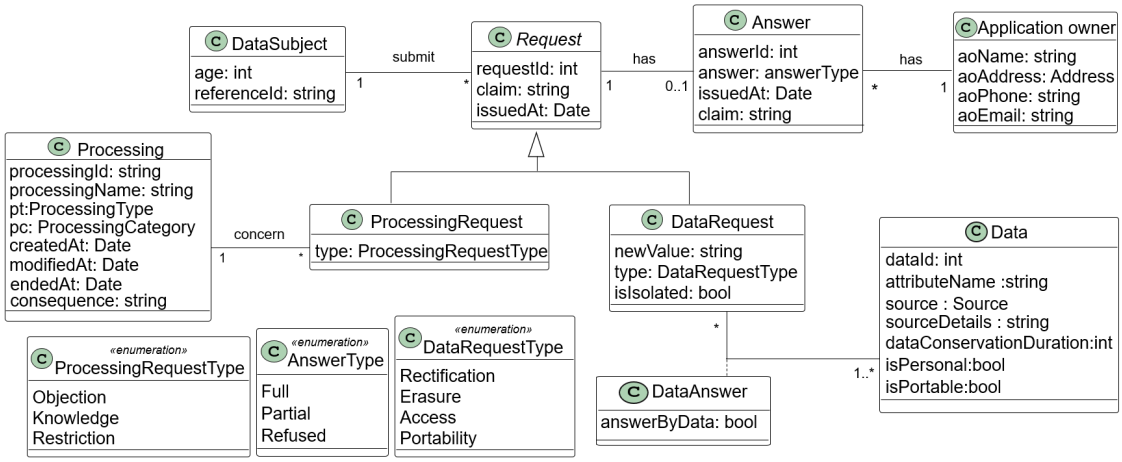

Figure 11: Rights package

of isolating (*i.e.*, forbidding the processing of) data while it is subject to rectification and until its accuracy is approved by application owners.

For example, `Sohan` notices an error in his data as recorded by the sport center: his `member_type` is mistakenly set to `"Silver"` (*see* Figure 4). `Sohan` sends then a rectification request, specifying both the data to be rectified (`member type`), and its new value (`"Gold"`) and claims `"My member type should be Gold instead of Silver due to a status update"`. If the `sport center owner` agrees and responds positively to the request, the new value will be automatically set in the sports center database.

[8]GDPR prescribes that response time be less than a month in the general case (and could be extended by two extra months in case the request is complex and the `ApplicationOwner` has to answer a large amount of `Requests`) (GDPR, Art 12.3).

4.1.6. Breaches package.
Figure 12 shows the `Breaches` package. `Breaches` are unplanned situations (*e.g.*, attacks, bugs) that hinder personal data security. When a breach occurs, GDPR prescribes `Breaches` management rules that revolve around a `Breaches` register.

`Breaches` must all be documented in detail. Their `nature`, their occurrence date (`createdAt` attribute of `Breach` class), their impact (`Consequence` class) and the `Measures` taken to remedy them are required to be logged.

In order to precisely define what piece of personal data is impacted, `Breaches` must also be linked to `Data` and `DataSubjects`. The number of affected `DataSubjects` is important to know too; it can be calculated.

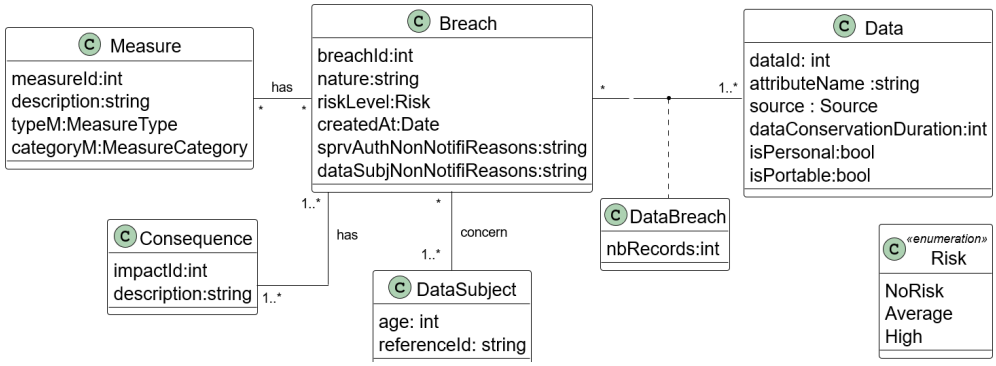

Figure 12: Breaches package.

When a `Breach` of `High` risk level is detected, the supervisory authority and the affected `DataSubjects` have to be informed. If the risk level is moderate (`Average` risk value) it is not mandatory to inform the supervisory authority and the affected `DataSubjects`. In that case, a justification for the lack of notification must be given (respectively using the `sprvAuthNonNotifReasons` and `dataSubjNonNotifReasons` attributes of the `Breach` class).

4.1.7. Documents package.
Figure 13 depicts the classes of the `Documents` package which gathers the various legal documents used in privacy management. There are four types of legal documents, relative to either `Processings`, `Consents` or `Breaches`:

- **Records of processing activities** (`RecordsProcessingActivities` class). Application owners must maintain a `RecordsProcessingActivities` as described in GDRP, Art. 30. The prescribed informations are available from the `Actors`, `Data` and `Processings` packages and automatically fed into this document.
- **Contracts** (`Contract` class). As explained previously (*see* `Consents` package), the `Contract` document of a `DataSubject`, lists all the `Processings` they consented to. It

specifies how personal data will be processed and what the purposes of `Processings` are.

- **Records of breaches** (`RecordBreaches` class). As explained previously (in the Breaches package), the `RecordBreaches` is key to both log data `Breaches` and notify the supervisory authority and affected `DataSubjects`. It contains detailed information on data breaches, including their nature or consequences.
- **Data Protection Impact Assessment (DPIA) report** (`DPIAReport` class). DPIA is a process to assess risks to individuals' privacy before conducting high-risk data processing activities (GDPR, Art. 35). A `DPIAReport` identifies potential risks and mitigation measures.

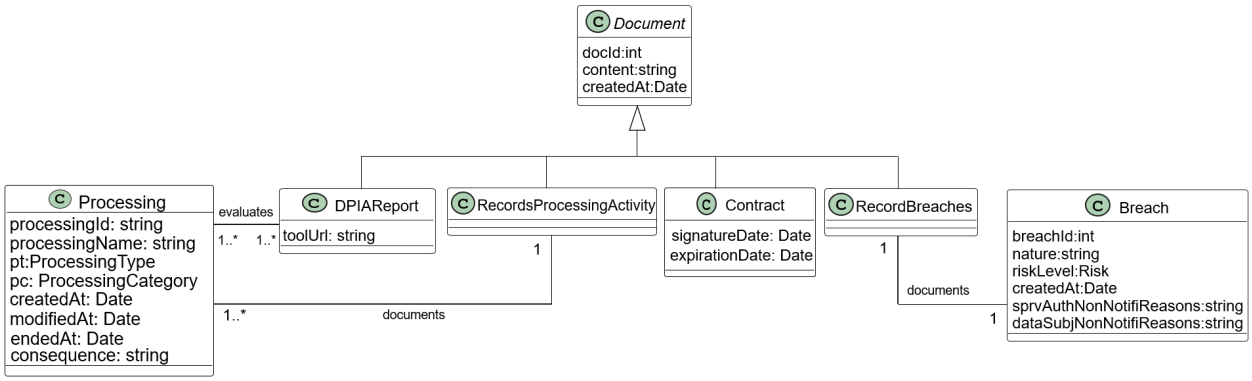

Figure 13: Documents package.

Records for processings activities, records for breaches, and contracts are automatically generated from the information available in instantiated classes of their respective related package. DPIA is applicable to specific processings (*e.g.*, that manipulate health-related data). This work, does not cover DPIA because it would necessitate to catalog all these specific cases and their associated data (*e.g.*, risks and their impact) for which dedicated tools already exist. The interested reader can refer to PIA tool [14], which offers a dedicated software tool and complementary documentation to assist this task. The URL of the tool is available for reference in the `toolUrl` attribute of the `DPIA` class.

4.2. Metamodel OCL Constraints

In order to provide a clearer view of the possible metamodel instantiations, a set of constraints defined in the Object Constraint Language (OCL)[9] is provided. These constraints precise PRIAM metamodel semantics. Also known as well-formedness rules, they ensure that each instance of the metamodel adheres to precise conditions.

[9]https://www.omg.org/spec/OCL/ [last consulted 2025-04-02].

For instance, within the `Actors` package, a `DataSubject` may be associated with either one `Tutor` or none (*see* association between `DataSubject` and `Tutor` on Figure 5). To ensure semantic integrity, it is necessary to precise the conditions under which a `DataSubject` is assigned a `Tutor`.

Twenty-five (25) constraints have been defined, each affecting a given metamodel package. The `Actors` package includes 5 constraints, the `Data` package contains 4, the `Processings` package embeds 5, the `Consents` package defines only 2, the `Rights` package 6 and the `Breaches` package specifies 3 constraints. The constraints were checked for OCL syntax and consistency with metamodel packages using Papyrus, an integrated development environment designed for modeling syntactically correct UML and SysML diagrams and OCL constraints[10].
To avoid unnecessary verbosity, only some constraint examples are provided[11]. For instance, the following two constraints align with Article 8 of GDPR:

- Constraint `C.A2` stipulates that, for a given country, the age under which a `DataSubject` is considered minor must be between 13 and 16.

```
context Country
inv: self.minorAge >= 13  and  c.minorAge <= 16
```

- Constraint `C.A3` stipulates that if a given `DataSubject` is considered minor in their country, they must have a `Tutor`.

```
context DataSubject
inv: self.age < self.country.minorAge implies not self.tutor ->
↪ isEmpty()
```

This section presented PRIAM metamodel which provides a structural view of privacy requirements as defined in the GDPR. A class diagram, split into seven packages, is completed by a set of twenty five OCL constraints that precise its semantics. Building upon this, next section will present the various artifacts that are derived from that metamodel and constitute the basis of the proposed privacy integration approach as sketched in Figure 1.

5. Derivation of Artifacts from the Metamodel

Section 4 has described how GDPR has been modeled into a UML metamodel with complementary OCL constraints. This section provides a description of all the additional artifacts (colored blue in Figure 1) of the privacy integration approach. These artifacts are produced using a model-driven engineering (MDE) process. These artifacts can be divided in three categories:

[10] `https://wiki.eclipse.org/Papyrus`
[11] The complete set of constraints is available online: `https://github.com/gitPriam/PRIAM_Metamodel/`

- application-independent artifacts which are produced at the Privacy Rights Domain Engineering stage and are directly obtained by transforming the metamodel information,
- application-dependent artifacts which are produced at the Privacy-aware Application Engineering stage that use information from the Application to transform and contextualize some of the previously produced application-independent artifacts, using annotations on the application as the source of application-specific data,
- and, user-specific artifacts which are produced at the Privacy-aware Application Operation stage that use information provided at runtime by end-users to log all the necessary information in order to manage end-user data privacy.

5.1. Application-independent Artifacts

As shown on Figure 1, there are three application-independent artifacts that are generated as companions of the PRIAM metamodel:

- the PRIAM DSL based on PRIAM metamodel,
- Generic user stories for GDPR requirements,
- and, the Privacy enforcement database schema.

Each of these artifacts will be described in this subsection.

5.1.1. The PRIAM DSL.

This subsection describes in more details the MDE process and PRIAM DSL. In practice, the MDE process is tooled by the **JetBrains Meta Programming System (MPS)** [15] which is one of today's most popular meta-programming environments. MPS offers a distinct approach compared to traditional textual MDE environments such as Eclipse / XTEXT. Like Eclipse / XTEXT, MPS allows users to define the syntax of a DSL. However, unlike Eclipse / XTEXT which relies on a DSL grammar, MPS uses a system called projectional editing. With MPS, developers interact directly with the model structure through a user-friendly, readable representation, without the need for a predefined grammar. MPS also comes along with a code generator to write transformations and a constraint expression language and checker.

MPS has its own metamodel that straightforwardly maps UML class model concepts. For example, a class becomes a concept, a simple attribute a property, and an association a reference. Figure 14 shows the definition of the `Data` class as an MPS metamodel instance.

To edit and manipulate MPS metamodel instances, MPS provides corresponding default GUIs called `Editors` that can further be customized. This customization provides a means to define a concrete textual syntax for the corresponding DSL. This is MPS's projectional

```
concept Data extends    BaseConcept
             implements INamedConcept

  instance can be root: true
  alias: data
  short description: <no short description>

  properties:
  dataId           : integer
  primaryKey       : boolean
  dataConservation : string
  source           : Source
  sourceDetails    : string
  isPortable       : boolean
  isPersonal       : boolean

  children:
  << ... >>

  references:
  personalDataCategory : PersonalDataCategory[0..1]
  dataSubjectCategory  : DataSubjectCategory[0..1]
```

Figure 14: `Data` class expressed in MPS.

```
<default> editor for concept Data
  node cell layout:
    [-
    Data
      attribute name: { name } is personal: { checkbox <<text>> {isPersonal} <no label> }
        is primary key: { checkbox <<text>> {primaryKey} <no label> }
      ? retention period: ?{ dataConservation } ? source: ?{ source } ? source details:
        ?{ sourceDetails }
      ? is portable: ?[> { checkbox <<text>> {isPortable} <no label> } <] ? category:
        ?( % personalDataCategory % -> { name } ) ? data subject category:
        ?( % dataSubjectCategory % -> { name } )
    -]

  inspected cell layout:
```

Figure 15: Editor customization definition for the `Data` MPS concept.

editing capability.

Figure 15 provides a the definition of a customized DSL editor example for `Data`. For example, the `isPortable` attribute of class `Data` appears as the `"is portable :"` label that precedes a checkbox to chose its `true` (checked) or `false` (unchecked) value as described in Figure 15. How such DSL editors are used by application owners to annotate `Data` from a given application is explained in Section 5.2.

5.1.2. Generic GDPR User Stories.
User stories are commonly used in software development as a means to describe requirements as short and simple scenarios that have to be implemented. These are usually gathered in the project backlog.

GDPR is a complicated legal document and the process of deducing application-specific requirements is not straightforward. This paper uses the proposed GDPR metamodel presented in Section 4 to list concrete application-specific user stories. They explicit requirements for developers to both understand what is expected by the regulation and have a comprehensive list of requirements so that none is forgotten.

User stories are produced in a two step process. First, generic user stories are provided that cover concepts of the metamodel. This first step is manual and fed with the expertise we have gained in deeply understanding the regulation. Second, generic user stories are contextualized to reflect application-specific information. This second step leads to a set of application-specific user stories that are integrated in the project's backlog. This second step is detailed in 5.2.

To describe generic user stories, we reuse the user story template introduced by Wautelet *et al.* [16].

`As a [who], I want [what], so that [why]`

where elements between `[ ]` have to be substituted by concrete information.

In order to be exhaustive, we used a systematic approach and translated GDPR requirements based on the package decomposition of the metamodel. The result is a set of 65 generic user stories. Table 1 shows two examples which respectively correspond to the `Processings` and `Rights` packages of the metamodel. The complete list of the generic user stories is available online[12].

It is important to note that the examples provided in Table 1 still have elements in between square brackets (*i.e.*, `[data subject]`, `[data list]` and `[processings list]`) . They each map to elements of the metamodel and have to be further substituted for, when the metamodel will be instantiated and contain application-specific information.

Table 1: Examples of generic user stories.

Package	**Generic User Story**
`Processings`	`U17: As a [data subject], I want to be informed of the legal basis for the processings on my personal data ([processings list])`
`Rights`	`U44: As a [data subject] I want to rectify my personal data ([data list]) so that I exercise my rectification right`

5.1.3. Privacy enforcement database schema.
The enforcement of GDPR rules strongly relies on a (privacy enforcement) database that stores application-specific information (*e.g.*, What are the processings of the application? or Is this specific data considered personal?) as well as user-specific information (*e.g.*, Does some specific user give their consent to this specific processing using this specific piece of personal data?). The structure of the database (its schema) depends only on the concepts gathered in the metamodel.

To generate the database schema, the metamodel is automatically transformed into a database schema creation script. This model-to-text transformation results in a SQL script that creates tables, relationships, and constraints. Figure 16 shows an excerpt from the generated script for the `Data` and `DataType` tables.
The database will then be populated with application-specific privacy enforcement data and user-specific privacy enforcement data in later steps described in Section 5.2 and 5.3 .

[12] `https://github.com/gitPriam/PRIAM_Metamodel/`

```sql
-- DataType table creation
create table data_type (
data_type_id int primary key,
data_type_name varchar(40));
-- Data table creation
create table data(
data_id int primary key,
data_name varchar(25), `source` varchar(25),
source_details varchar(255),
data_conservation_duration int DEFAULT -1,
is_personal boolean,
is_portable boolean,
is_primary_key boolean,
data_type_id int,
personal_data_category_id int,
data_subject_category_id int,
foreign key (data_subject_category_id) references `priam-actor`.data_subject_category(data_subject_category_id),
foreign key (data_type_id) references data_type(data_type_id),
constraint check_source check (`source` in('DIRECT','INDIRECT','PRODUCED')),
foreign key(personal_data_category_id) references personal_data_category(personal_data_category_id));
```

Figure 16: Excerpt of the generated Privacy enforcement database schema

5.2. Application-specific Artifacts

5.2.1. DSL-powered metamodel instance.

Starting from editors such as the one defined in Figure 15 for `Data`, the application owner can fill all the necessary application-specific information. An example would be specifying that the `Gender "attribute name :"` in the `sport center` application is considered to be personal data (`"is personal:  "` is checked). This is what we call an annotation in our work.

Figure 17 is the result of annotation for two examples of `Data`. It shows two instances of the `Data` concept: the `f(amily)Name` and `gender` of a `member` of the `sport center`.

It is important to notice that annotations are syntactically correct by design as editors prescribe most of the text written using the DSL. Type-checking is supported and traditionally powered type-based completion mechanisms limit type errors. Annotations are not only checked against their syntax though. Semantic checks are also performed. Indeed, constraints (as defined in 4.2) are evaluated. Their violation generate precise error messages that explain remedy strategies based on GDPR rules. For example, if the `source` of a given `Data` is `Direct`, data portability obligations apply: `isPortable` should be set to `false`. If not, an error is displayed that explains the semantic incompatibility with GDPR rules : `"All direct source data is portable"`.

Reaching the application's full annotation typically involves annotating its instances of `Data`, `Processings` and `Actors`. When the application (*e.g.*, the `sport center` application) is fully annotated using the DSL, the result is the DSL-powered metamodel instance artifact.

Furthermore, these annotations are in turn used to automatically generate:

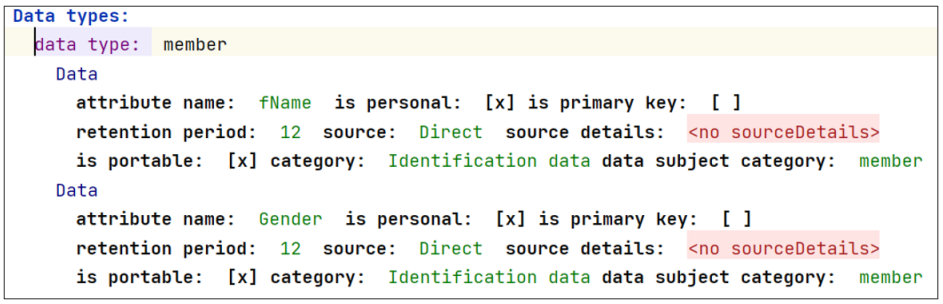

Figure 17: `Data` annotation examples obtained from using the (editor of the) DSL.

- the **Specific user stories backlog** that gathers Specific User Stories. Those Specific User Stories results from automatically injecting application-specific information originating from DSL annotation into Generic User Stories, as presented in Section 5.1,
- **Application-specific privacy enforcement data** that results from automatically injecting application-specific information from DSL annotation into SQL scripts that populate the a database the schema of which is presented in Section 5.1.

Both artifacts are described and illustrated in the following.

5.2.2. Specific user stories backlog.
In order to constitute a backlog targeted at application developers that contains all GDPR requirements declined specifically for an application, a specific user stories backlog is provided. Backlogs such as this one are considered to be the cornerstone of modern development practices. The backlog contains application-specific user stories that are clear and well-defined requirements. For `Application developers` to fully integrate GDPR into their applications, the MDE process uses generic user stories as introduced in 5.1 and annotations in order to derive specific user stories tailored to the context of the application. Details of `Actors`, `Data` and `Processings` are injected into Generic User Stories to form Application-Specific User stories. This injection is the result of an automatic text-to-text transformation. To illustrate this, let us refer to Generic User Story `U44` from Table 1:

`As a [data subject] I want to rectify my personal data ([data list])`
`so that I exercise my rectification right`

Based on the annotations made with the DSL, where `DataSubjects`, `Data` and `Processings` are known, two specific user stories are derived:

`As a (sport center) member, I want to rectify any of my personal data`
`(fname, age, password, member_type, gender, email), so that I exercise`
`my rectification right`

`As a trainer, I want to rectify any of my personal data`
`(name, specialization, contact_number),`
`so that I exercise my rectification right.`

Such specific user stories provide clear guidance to application developers. They provide detailed application-specific requirements that are easier to fulfill than more abstract requirement specifications would be, thus saving time, by avoiding unnecessary iterations, and increasing quality and coverage, by lessening the potential amount of errors.

5.2.3. Application-specific privacy enforcement data.
Once PRIAM DSL-based annotations are provided in the form illustrated by Figure 17, they are automatically translated by a text-to-text transformation so as to populate the database. The result of this transformation is a SQL script as illustrated by Figure 18.

```
/* insert data types and data */
insert into DataType(data_type_id, data_type_name) values (1, 'member');

insert into Data(data_id, data_name, source, source_details, data_conservation_duration, is_personal,
            is_portable, is_primary_key, data_type_id, personal_data_category_id, data_subject_category_id)
   values (2, 'fName', 'Direct', '',12,true,true,false,1,4,2);

insert into Data(data_id, data_name, source, source_details, data_conservation_duration, is_personal,
            is_portable, is_primary_key, data_type_id, personal_data_category_id, data_subject_category_id)
   values (3, 'Gender', 'Direct', '',12,true,true,false,1,4,2);
```

Figure 18: Excerpt of a SQL script for populating the database with application-specific enforcement data

This constitutes the application-specific privacy enforcement data. It reflects both:

- the nature of the information that is specifically dealt with in the application *i.e.*, its specific `Data` and `Processings`. For example, Figure 18 shows the insertion of a `Gender Data` that relates to the `member DataType`.
- and the information relative to how it should be considered with respect to GDPR. For example, Figure 18 shows the `Gender Data` is considered personal as the 6^{th} column for the `Gender` record that corresponds to the `isPersonal` annotation has the `true` value.

This application-specific privacy enforcement data is the last step before the privacy aware application runtime where user-specific privacy enforcement data is collected and used to concretely manage GDPR enforcement for specific application `End-users` (aka `DataSubjects`).

5.3. User-specific Artifacts
User-specific privacy enforcement data is the information that is recorded in order to be able to enforce end-users' privacy rights during the **Privacy-aware Application Operation** stage of Figure 1. It typically involves `DataSubjects`, `Consents`, `Rights`, `Breaches` and `Documents` and corresponds to the runtime of the privacy-aware application. This information is provided by `End-users` via a dedicated privacy management GUI[13] that is added

[13]The detailed description of this GUI and of the micro-services, grayed in Figure 1, is beyond the scope of this paper.

to the application in order to provide them with privacy enforcement capabilities.

Let us recall the example of `Sohan` (*see* Rights in page 18) that wants to rectify his `member_type` in the `sport center` application from the `"Silver"` to the `"Gold"` value. To do so, `Sohan` uses a dedicated GUI where a `Rectification` form is shown. This form lists the `DataNames` of his personal data from which he can chose from. These `DataNames` come from the Application-specific privacy enforcement data and are sorted by `DataSubjectCategory` (*i.e.*, by `member` in this specific example). `Sohan` selects the concerned `DataName` (*i.e.*, `member`) and provides both its new `"Gold"` value and the claim `"My member type should be Gold instead of Silver due to a status update"`. This `Rectification DataRequest` information (together with its `issuedAt` date) constitutes `Sohan`'s User-specific privacy enforcement data.

This information is recorded in the database using a SQL script as illustrated by Figure 19. Now that the metamodel has been presented and its practical application through its implementation as a DSL and the generation of application-independent, application-specific and user-specific artifacts demonstrated, next section discusses the experiments conducted to evaluate the metamodel.

Figure 19: Excerpt of a SQL script for populating the database with user-specific enforcement data

6. Metamodel Evaluation

In order to evaluate the proposed metamodel, we conducted a set of experiments aimed at answering the following research question:

RQ: Do the proposed metamodel and its associated user stories translate appropriately GDPR requirements into technical / processable concepts?

We used two complementary methods for answering RQ:

- **Global evaluation via a questionnaire**. To evaluate the pertinence of the metamodel, we used a standard "check-list" that is used in practice for verifying the compliance of a technical solution with GDPR. The choice of a questionnaire is justified by its practicality, making it possible to avoid submitting the entire text of the regulation to experts, while relying on validated tools from literature, designed to assess GDPR compliance.

- **Element-by-element evaluation**: to complete the evaluation conducted through the first method, we took each element in the metamodel and its associated artifacts, and checked if it is addressed in at least one article in GDPR text. We leverage Generative AI for conducting this part of the evaluation.

6.1. Method 1: Global Evaluation with a Questionnaire

We first present how we designed the questionnaire used in this evaluation. Then, we expose the process for its completion and analyze the results.

6.1.1. Questionnaire Design

To set up our questionnaire, we decided to use a questionnaire designed and tested by experts in the field. After a review of the literature, we identified several relevant questionnaires [17, 18, 19, 20, 21]. We observed that, while questions were often grouped into categories, these categories varied and could not be directly compared. Instead, we analyzed each question individually and identified the GDPR aspects it addressed. A set of 24 aspects have been listed. We then grouped these aspects into five overarching classes used as comparison criteria:

- Lawfulness (consent, minors' consent, legal basis, records of processing)
- User rights (information, access, portability, erasure, rectification, objection, etc.)
- Accountability & governance (purpose, accuracy, minimization, retention, third parties, DPO, DPIA, representative)
- Security and breaches
- Data transfer outside EU

Our qualitative analysis showed that the questionnaire from dataprotection.ie [17] covered most of the aspects (22 out of 24).

To confirm these results, we conducted a quantitative NLP-based assessment. We compared each questionnaire with two GDPR reference sources:

- A glossary of 430 French-English GDPR terms from CNIL (National Commission on Informatics and Liberty – the data protection authority for France)
- The full text of the GDPR (99 articles)

Our methodology included:

- Collecting and preprocessing questionnaire texts (tokenization, stop-word removal, lemmatization)
- Vectorizing documents using embedding models (Word2vec and FastText)
- Measuring similarity with cosine similarity

Results confirmed our qualitative findings: the questionnaire from dataprotection.ie achieved the highest similarity scores with both GDPR key terms and full text, followed by the questionnaire from [21]. Both embedding models produced consistent results.

We thus selected the questionnaire from dataprotection.ie [17] as the basis for our work, given its broad coverage and highest similarity to GDPR.

Detailed information about the process and the complete results are available in the following Git repository: `https://github.com/gitPriam/PRIAM_Metamodel`

6.1.2. Questionnaire Completion

Once the questionnaire was selected, we tailored it to our study's requirements while preserving its core essence. In particular, we modified the answer proposals and added questions to ask the respondent some extra information about the given answer. Since the goal of the evaluation is to know if PRIAM metamodel meets the different requirements of GDPR, there were four answer options:

1. `fully modeled`: the metamodel adequately meets the requirements presented in the question.
2. `partially modeled`: a part of the question has been considered by the metamodel.
3. `not modeled`: the metamodel has not covered the question.
4. `should not be modeled`: the question raises an organizational and not a technical / processable requirement. It should not be considered in our metamodel.

In case the answer is different from `fully modeled`, we ask the participant to justify it. When the answer is `fully / partially modeled`, we collect which part of the metamodel takes into account the GDPR requirement considered by the question (single choice answer). Finally, the perceived quality of the questionnaire is assessed by asking expert participants to evaluate the relevance of each question (through a `yes / no` answer).

6.1.3. Validation by the experts

For this study, we target people with expertise or knowledge in the field of data privacy. We contacted potential participants by email, including specialized mailing lists known to DPOs. We then expanded our outreach by sharing the call for contributions on professional networks, notably LinkedIn. They were asked to view the videos presenting the different parts of the metamodel before responding[14]. We selected 4 experts with different profiles in IT (data management) and Law domains. In order to ensure that we were collecting opinions from persons with real expertise on the subject, we asked the participants to self-assess their knowledge on GDPR (legal knowledge on GDPR) as well as on its implementation (IT knowledge on GDPR implementation) on a scale of 5. Table 2 shows the job title, the number of years of experience on privacy issues and the self-evaluation of each expert.

[14] The questionnaire, videos and experts' answers are available online: `https://github.com/gitPriam/PRIAM_Metamodel/`

Table 2: Overview of job title, GDPR knowledge assessment and experience of participants.

	Job Title	GDPR: legal	GDPR: IT	experience (yrs)
Expert 1	DPO & legal researcher	5	4	7
Expert 2	DPO & Doctoral Candidate in Law	5	4	5
Expert 3	GDPR Correspondent & Research / Open Data Manager	3	1	2
Expert 4	DPO	5	4	5

According to their self-assessment, experts 1, 2 and 4 have the same level of knowledge about GDPR, both in terms of the legal text and its implementation. However, expert 1 has 2 years more experience. Expert 3, on the other hand, has a background in IT, knowledge of GDPR and 2 years of experience in the field.

6.1.4. Analysis of results

Table 3 presents the average percentage by response option. Percentages are given here just to highlight the high or low ratio of respondent answers for each response option. These results show that experts assess that our metamodel covers most of the questionnaire as approximately only 8% (7% + 1%) questions are judged `not modeled` or `should not be modeled`.

Table 3: Average percentages by response option.

Experts	Fully (%)	Partially (%)	Not modeled (%)	Should not be (%)
expert 1	96	0	4	0
expert 2	54	33	13	0
expert 3	83	15	0	2
expert 4	83	5	10	2
Average	**79**	**13**	**7**	**1**

By analyzing the arguments of the experts for the answers `partially modeled` / `not modeled`, we observed that these relate to organizational requirements or to functional requirements implying the specification of business logic, beyond a structural formalization as a metamodel. Examples are given in Table 4. For instance, age verification is not actually represented in the metamodel, although it keeps track of the age of users as well as the minor age for countries.

To the question "This question has been dealt with in (which) part (?)", linking questions with the packages of the metamodel, we measure on average, a correspondence of 75% between the answers of the DPOs and our intention. For expert 3, this correspondence drops to 48% (which can be explained by their lack of experience in IT). These results were obtained by considering all responses, including null values (in `not modeled` and `should not be modeled` cases). When ignoring the null values, averages are 82% and 49% respectively. Moreover, experts consider 100% of the questions to be relevant. Finally, in response to the open question "`What do you think of this questionnaire?`", 2 out of 4 experts found the questionnaire complete.

Table 4: Examples of justifications for not fully modeled questions

Question	Justification
Where online services are provided to a child, are procedures in place to verify age and get consent of a parent / legal guardian, where required?	How to check the real age?
Are there procedures in place to notify data subjects of a data breach (where applicable)?	This depends on organizational constraints

6.2. Method 2: Global Evaluation with an LLM

To complement expert evaluation, we have implemented an approach leveraging large language models (LLMs). This approach involves submitting the same questionnaire used for expert evaluation to an LLM[15].

6.2.1. Input & output files

In this validation, the LLM takes as input the questionnaire and the metamodel.The latter is organized into distinct packages, each containing:

- the description of each meta-class, including its attributes, in natural language,
- the associations between meta-classes, represented with the textual syntax of PlantUML [22], which is close to natural language and thus exploitable by LLMs,
- the associated constraints and generic user stories. To ensure accurate interpretation by the LLM, the original OCL constraints are translated into natural language.

The process involves asking the same questions to the LLM as to the experts, ensuring comparability for the evaluations. For each question, the LLM is tasked with providing a response and a justification. If the answer is `fully` or `partially modeled`, the LLM must also identify the relevant part of the metamodel.

6.2.2. Prompting

In this section, we explain the prompting techniques used to optimize the results of the LLM [23]. Initially, we opted for a zero-shot prompt [24], providing a prompt equivalent to the introductory text given to the experts, with a few exceptions, such as the removal of the text inviting the experts to view the explanatory videos. The prompt is structured as follows: the goal of the validation, the description of the response options, the questions of the questionnaire and the description of the metamodel. Despite clear definitions of the response options, several issues emerged during the first executions:

- Justifications for responses. The model tended to include organizational or procedural aspects in justifications, without choosing the `should not be modeled` answer,

[15]All files, code, prompts, and results related to this study are available in the repository.

although the LLM has been clearly instructed to consider these concerns as out of the scope of the metamodel.

- Finding explicit concepts. The model only checked for the presence of concepts explicitly mentioned in the question.
- Responses based on preexisting knowledge. The LLM used its general knowledge to answer the questions, not considering only the metamodel elements included in the prompt. This led to incorrect and irrelevant results.

To address these issues, we adjusted our prompt by adopting the Chain-of-Thought (CoT) technique [25], explicitly guiding the LLM to decompose its reasoning into sequential steps. Furthermore, we provided explicit instructions to the model and specified an output format to ensure consistent and structured results. The main changes are as follows:

- Specification of reasoning steps: introducing reasoning steps to guide the model through a detailed analysis before reaching a conclusion.
- Adjustment of response option definitions: clarify the definitions and application of response options.
- Precisions on the `should not be modeled` option: providing a stricter definition of the aspects to be ignored in the evaluation, particularly those related to organizational or procedural aspects.
- Other Instructions: (i) Consider only the content of the metamodel. (ii) Perform an in-depth analysis of the metamodel, taking into account the semantics of meta-classes, meta-associations, meta-attributes, and constraints. (iii) Strictly adhere to the output format.

The evaluation is conducted by evaluating each question with the entire metamodel, rather than with individual packages. This choice was made to guarantee full contextual consideration for each question while complying with the token limit. The instructions therefore emphasized the importance of considering the entire metamodel, to avoid partial or incorrect results caused by limited information provided to the LLM.

6.2.3. Model selection and performance optimization

For the evaluation, we selected the Mistral AI LLM [26] and selected the "mistral-large-latest" model, which offers open access to its API. This choice is based on the promising results obtained in the initial tests, leading us to adopt it for the entire evaluation process. To ensure an optimal use of the model, several adjustments were required:

i. Temperature setting. Temperature is a crucial parameter controlling randomness in model performance, allowing users to balance coherence and creativity in text generation[16]. For this task, the temperature was set to 0 to favor deterministic and consistent responses. This setting is particularly important in a legal context where accuracy and consistency are paramount.

ii. Error management. Another key aspect of the process is error management, which can be caused by external constraints such as temporary API overload or API rate limits. To handle these errors, we implemented an exponential backoff strategy [17], which automatically retries failed requests with progressively increasing delays between attempts.

iii. Individual question processing. Each analyzed question is submitted individually, with its complete context (objective, instructions, and metamodel description). This approach guarantees independent processing for each question, avoiding potential influences or interferences that might arise when requests are processed as batches.

6.2.4. Baselines

To assess the reliability of the results obtained by the LLM, we conducted several baseline tests, with various definitions of the metamodel. These tests aim to demonstrate that the LLM's responses are consistent, relevant to GDPR context, and based only on the provided metamodel.

Empty Metamodel. This test verifies that, as explicitly requested in the prompt, results depend on the information provided to the LLM rather than on preexisting knowledge or random assumptions. With an empty metamodel as input, the LLM should not return any `fully` or `partially modeled` question.

Out-of-Scope Metamodel. We aimed to confirm that the LLM does not provide generic or superficial responses, but relies on the provided context to generate accurate answers. This underlines the importance of providing a metamodel adapted to the context of the regulation. With a metamodel about animals as input, the LLM should again not return any `fully` or `partially modeled` question.

Partial Metamodel. We tested the LLM with a partial metamodel, retaining only the `DPO` and `Country` classes. This test was designed to observe how the LLM handles incomplete information and to ensure that it does not make arbitrary associations between elements, which could lead to partially correct answers.

Table 5 shows the results of the questionnaire evaluation for each baseline case and compares them with the PRIAM metamodel.

[16] `https://docs.mistral.ai/guides/sampling/#temperature`
[17] `https://cloud.google.com/memorystore/docs/redis/exponential-backoff`

Table 5: Evaluation of Questionnaire by Mistral Large model (Baselines and PRIAM Metamodel)

Metamodel Version	**Fully Modeled**	**Partially Modeled**	**Not Modeled**	**Should Not Be Modeled**
Empty Metamodel	0%	0%	96.30%	3.70%
Metamodel Out Of GDPR Scope	0%	0%	94.44%	5.56%
Metamodel with Partial GDPR Coverage	7.41%	12.96%	79.63%	0%
PRIAM Metamodel	**83.33**%	**16.66**%	0%	0%

In the first two evaluations (*i.e.*, Empty Metamodel and Metamodel Without GDPR Scope), as expected, no questions were classified as `fully` or `partially modeled`. In the `Empty Metamodel` baseline, a few elements (3.7%, 2 questions) were classified as `should not be modeled` while in the `Metamodel Out Of GDPR Scope` baseline, 5.56% (3 questions) were classified as such. These elements are related to organizational aspects and specific contexts that can be confusing for the LLM. For example, questions like *"Are plans and procedures regularly reviewed?"* relate more to internal management and governance processes rather than direct GDPR requirements.

It is important to note that in some runs, we obtained 100% of questions classified as `not modeled`. This phenomenon is attributed to the design of LLMs, which operate using stochastic schemes. Despite setting the temperature parameter to 0 to minimize variability, LLMs remain inherently non-deterministic, leading to slightly variable results across runs. However, results are overall consistent and relevant, when analyzed in light of the justifications provided by the LLM.

In the third evaluation, which involves a partial metamodel with only two classes (`DPO` and `Country`), the table shows a higher percentage of questions `fully` or `partially modeled`. This outcome is expected, as the metamodel aligns more closely with some evaluated questions. For example, the questions related to the DPO (questions 30-33) are classified as `fully modeled` or `partially modeled`. Other questions, such as question 52, which relates to data transfers, are classified as `partially modeled`. This is understandable, as the metamodel includes the concept of Country but lacks key elements to fully model data transfers, such as data categories and processing purposes. The LLM's justification for this classification is:

"The question asks whether all transfers are listed, including details such as the nature of the data, the purpose of the processing, the countries involved in the data export and import, and the recipient of the transfer. The metamodel includes the 'Country' class, which can represent the countries involved in the data transfer ('countryId', 'name', 'adequate'). However, it lacks classes or attributes to represent the nature of the data, the purpose of the processing, and the recipient of the transfer. These are key concepts that are missing or incomplete in

the metamodel."

Similarly, the classification for question 5, which relates to consent for minors, was correctly justified by the LLM. This demonstrates the accuracy of the LLM in associating precisely the classes of the metamodel with the questions.

Through these three evaluations on baselines, we have validated that the model can accurately respond to the questionnaire. This forms a solid foundation for cross-validating the PRIAM metamodel.

6.2.5. Results

With the full PRIAM metamodel, 83% of the questions are classified as `fully modeled` and 16% as `partially modeled` (*cf.* Table 5). The LLM justifies partial responses by indicating that explicit representations and specific mechanisms are missing in the metamodel. For example, for the question: *"Are industry standard encryption technologies employed for transferring, storing, and receiving individuals' sensitive personal information?"*, the justification for the partial response is as follows:

"However, the metamodel lacks explicit representation of encryption technologies and the processes of storing and receiving data. While the Measure class can imply encryption as a measure, it does not explicitly model encryption technologies or the specific processes of storing and receiving data."

This justification is valid and coherent with the scope of the metamodel, which is not intended to encompass organizational requirements, business logic, or some implementation-level technical elements.

When comparing the results produced by the LLM with those obtained from the experts, a close correlation is observed: 83% of the responses are classified as `fully modeled` by the LLM, as compared to 79% by the experts. 16% are classified `partially modeled` by the LLM, as compared to 13% by the experts. Additionally, the justifications provided by the LLM align with those from the experts.

This demonstrates that the PRIAM metamodel effectively covers the questionnaire and provides relevant concepts for implementing the software requirements derived from the GDPR.

6.3. Method 3: Element-by-element Evaluation

This experiment aims to verify that the PRIAM metamodel, including its constraints, and the derived list of user-stories contain only relevant elements to formalize the requirements of GDPR.

6.3.1. Input & output files

To avoid bias and ensure in-context analyzes, we conduct separate evaluations for each different element. The output for each validation is simple: a "Yes" or "No" regarding the alignment of the element with GDPR, accompanied by a justification. When the element is considered relevant, the LLM specifies the corresponding chapters and articles of the GDPR. Otherwise, an explanation states why the element does not match any requirements.

6.3.2. Prompting

For this experiment, we adopt a Zero-Shot structured prompt. The LLM is provided with the context, a detailed description of the element to be evaluated, the output format and clear instructions to follow. The instructions (based on Instruction-based prompting) guide the model to focus solely on the provided element description and to evaluate its compliance with GDPR chapters. The model is also instructed to generate clear and concise responses, formatted as requested (which falls under Format-constrained prompting).

6.3.3. Model selection and performance optimization

Given the results obtained in the previous evaluation with Mistral AI model, we chose to use the same model, with the same parameters and prompt adjustments. These elements again demonstrated their relevance and effectiveness in optimizing the performances of the model.

6.3.4. Results

Table 6 presents the results of this evaluation. All generic user stories are classified as related to GDPR. The justifications generated by the model are precise and relevant.

Let us consider for instance the user story:

"As a data controller, I want to document the contact details of data consumers (third-party recipients and processors)."

The model proposes this justification:

"This user story complies with the requirements of Chapter IV of GDPR, specifically Article 30, which outlines the obligations of controllers and processors regarding records of processing activities."

In the subsequent evaluation, it is observed that 20% of the constraints (5 out of 25) were classified as non-aligned with GDPR. Three of these constraints enforce consistency for dates, implementations details which are not explicitly mandated by GDPR. The remaining two constraints are:*(i)"There must be exactly one main purpose in the list of purposes associated with data processing". (ii) "If the data source is Generated, then the data must not be portable"*. These two constraints have been added to apply recommendations proposed by CNIL, the administration in charge of enforcing the application of GDPR in France. For the former constraint, CNIL provides examples where processing can have secondary purposes,

derived from a main purpose[18]. For the latter constraint, CNIL gives precisions about the management of portability (Article 14 of GDPR) for data derived from personal data, such as those generated by user profiling[19].

Regarding the alignment of class descriptions (and their attributes), only one class was classified as not-aligned: DataType. This classification is accurate, as this concept is not explicitly mentioned in GDPR. It has been included in our metamodel for operational purposes, as a means to locate personal data in the code of applications.

Table 6: Alignment of different artifacts categories with GDPR requirements.

Category	**Aligned with GDPR (%)**	**Not Aligned with GDPR (%)**
Generic User Stories	100%	0%
Constraints	80%	20%
Classes & Attribute Descriptions	97.05%	2.94%

This evaluation confirms that metamodel elements and all user-stories are relevant to formalize the software requirements implied by GDPR. The LLM has shown a strong capability in accurately assessing compliance, as evidenced by the 100:% alignment ratio for generic user stories. Furthermore, the model's justifications and its ability to distinguish between concepts directly related to the regulation and those that are not underscore its effectiveness.

6.4. Threats to Validity

Despite our methodical approach and the rigor with which we designed and elaborated our evaluations, there are still some potential threats to validity regarding the obtained results.

Internal Threats.

The relevance and completeness of the questionnaire are primordial for a reliable evaluation of the metamodel. To assess these qualities, we applied a rigorous methodology, analyzing all 154 questions from five questionnaires proposed in the literature, extracting a comprehensive set of concepts, and finally selecting the most complete questionnaire. A second validation, based on semantic similarity measures, confirmed the selection of the questionnaire. Finally, as a cross-validation, the experts who evaluated the metamodel were asked to provide feedback on the questionnaire. This feedback was also positive about its relevance and completeness.

The length of the selected questionnaire, compared to other questionnaires in the literature, is important and may be considered a potential threat to the validity of the results. A long questionnaire can demotivate participants and lead to inattentive and / or random responses.

[18] `https://www.cnil.fr/sites/cnil/files/atoms/files/record-processing-activities.ods`

[19] `https://www.cnil.fr/fr/professionnels-comment-repondre-une-demande-de-droit-la-portabilite`

To mitigate this potential threat, we gave participants the option of saving a partially completed questionnaire, i.e., they could return to the questionnaire at any time without losing their previous answers. The participants were also informed that their answers were very important (valuable) and contributed to a scientific research work. The results presented in Section 6 demonstrate the serious commitment of the participants, who answered the entire questionnaire exhaustively.
Another threat to internal validity is the use of generic LLMs, without any retraining for the rather specialized software engineering tasks we request in our prompts. There is then no guarantee that the LLM correctly interprets our instructions and, for instance, limits the information and knowledge used in its analyses to the input and the context we provide. To alleviate this issue, the LLM is prompted to generate justifications for all the responses and we checked that these justifications always refer relevantly to elements of the metamodel or of the GDPR. Moreover, to reduce the risk of interferences, each question was processed separately.

External Threats. The number of participants in our study may be a limitation in terms of external validity, as the generalizability of the results may be affected. However, it is important to note that all four participants came from the field and have a deep understanding and knowledge of the topic. Three of them are DPOs and have been working on real cases for 5-7 years. The third has an IT background and is a GDPR correspondent for research / open data. Therefore, even though the sample size is small, their expertise adds significant value to the collected data.
Moreover, we cross-validated the evaluations of the experts with an LLM-based evaluation, to address the potential biases of subjective interpretations.

Conclusion Threats. LLMs predict tokens with associated probabilities, so interpretation can differ from run to run, leading to variable results. To reduce these risks, the temperature parameter was set to its minimum value, 0, to make the LLM as deterministic as possible.

Construct Threats. LLMs may not rely solely on the provided PRIAM metamodel description to evaluate the questions and mobilize its general knowledge of the regulations. In this case, the assessment extends beyond the defined framework to a broader interpretation, potentially introducing biases and deviations from the specific criteria to be analyzed. Baseline tests were used to detect such deviations and to adjust prompts (*e.g.*, by adding explicit instructions and structured output prompting).

7. Related Work

Maintaining confidentiality and securing personal data is crucial in today's digital age. A great deal of research has gone into preserving end-user privacy, and enhancing the level of confidentiality and security of personal data managed and stored in applications. Toumia *et al.* [27] propose a collaborative ontology named ColPri to represent the configuration of user privacy policy in IoT applications. Users can represent consents to exploit their personal

data and withdrawal. Pape *et al.* [28] propose a privacy solution based on Fog computing to avoid centralization and thus limit access to personal data in the cloud. Benhamida *et al.* [29] also propose an architecture based on Fog computing. This solution focuses on managing users' preferences when it comes to sharing their sensitive personal data. Gharib *et al.* [30] propose an ontology related to user privacy requirements to avoid privacy breaches during application design. Danilo M. Nogueira *et al.* [31] propose a personal data management model for transparency and consent management in Smart Personal Assistants.

Although these works target privacy-related problems by basing their solutions on frameworks, privacy models or ontologies, compliance with the fundamental principles of GDPR and user-oriented requirements is beyond their scope. In this paper, we focus specifically on how to model and implement GDPR principles.

To address the various challenges posed by GDPR, European initiatives such as SMOOTH GDPR [32], DEFeND [33], PoSeID-on [34], Papaya [35], PDPE4 [36] and BPR4GDPR [37] offer dedicated platforms. Each project takes a distinct approach to data protection challenges, intervening at different phases of the development lifecycle and employing various organizational and technical strategies. For example, SMOOTH focuses on monitoring micro-entreprises compliance and diagnosing GDPR-related anomalies, while Papaya acts as a subcontractor, outsourcing data analysis while ensuring data confidentiality. PoSeID-on employs privacy-protecting security techniques such as blockchain, complemented by a dashboard for users to control their personal data. DEFeND follows a modular strategy, ensuring flexibility to meet the organizational and technical needs of customers. BPR4GDPR offers a re-engineering process, providing a set of tools and methodologies for GDPR compliance. Although the general goals of these works are clear, the concrete details remain difficult to understand due to a lack of open-access resources describing the solutions and their implementations.

Despite differences in technologies and objectives, these platforms often provide integrated cloud-based solutions. Some may use third-party service providers to store users' personal data (*e.g.*, DEFeND) or request access to confidential documents for compliance verification (*e.g.*, SMOOTH). The platform developers emphasize compliance with regulations, including the deletion of documents after verification. In contrast to these initiatives, we propose a model-driven method that targets application owners and developers to guarantee personal data protection.

In the following, we focus on works that aim to address GDPR compliance problem. The focus is on works whose solution is based on models that simplify the representation and understanding of GDPR, for example by proposing models, ontologies, taxonomies or even translating GDPR into a set of requirements. We classify these works according to the following criteria: *Work Goal* which refers to the final objective of the research work. It takes as values: (i) Requirements Modeling and elicitation (rm) and (ii) Checking Compliance

(cc). *GDPR Formal Representation (MOD)* defines the conceptual model that the authors used for the formal representation of GDPR concepts. *GDPR Modeling Coverage (MOD-C)*[20] depicts if the given representation covers some or all of the criteria presented in the `qualitative evaluation` subsection. *Requirements (REQ)* criterion exposes whether the work cites/lists GDPR requirements needed to develop a solution that ensures compliance with this regulation. *Requirement Coverage (REQ-C)* states if the cited work fully or partially covers requirements. *Support (SUP)* column indicate whether the authors proposed a tool to support GDPR. *Experts intervention (EXP)* column indicates whether lawyers participate in the development or validation of the work. Table 7 compares the 19 articles we analyzed[21]. Based on the "Work goal" criteria presented in Table 7, two categories of works have been distinguished.

Requirements Elicitation and Modeling. To represent the requirements of GDPR, some works have proposed to simplify the reading and understanding of this regulation to non-specialists by extracting the requirements. However, the way in which they are represented differs. Some works are based on a metamodel representation. Diamantopoulou *et al.* [38] present a metamodel that captures some of the concepts of GDPR. It focuses on privacy level agreements to support privacy management. Their metamodel is specific to public authority services (e-service) and does not show the practical transposition to development processes.

Tom *et al.* [39] also propose a formal representation of GDPR simplifying the representation of this law. Two sub-metamodels were designed. The first includes GDPR entities. The second represents users' rights. The authors intend to use this model as a reference tool for defining an application's organizational privacy policy. Burneister *et al.* [4] introduce a GDPR extension to TOGAF Enterprise Architecture reference metamodel [40]. The proposed representation is closely linked to enterprise architectures and remains conceptual (not tooled). No details about the practical implementation of the metamodel are provided.

Caramujo *et al.* [41] propose a DSL called RSL-IL4PRIVACY, which is a domain-specific language specifically designed for privacy policies. This language aims to specify and document these policies in a structured manner. To achieve their goal, the authors present a metamodel that represents the key concepts (Statement, Recipient, PrivateData, Service, and Enforcement) in a privacy policy. This abstract representation served as the basis for their DSL. To support the language, the authors implemented a tool (RSLingo4PRIVACY-studio) that enables privacy policies to be represented and transformed into other representations

[20] A detailed table is available on the repository.

[21] A keyword search (General Data Protection Regulation, GDPR and Privacy) was carried out on Google Scholar, IEEE Explore, ACM and Springer search engines. More specific terms such as MDE, Requirements Modeling, Compliance, Data Protection, Conceptual Modeling were then added and combined (e.g. of combination, GDPR and Requirements Modeling). The selection of articles was based on abstracts and the presence of a representation (metamodel, ontology, etc) of the European regulation

such as Word and Excel.

Other works have chosen to use ontologies. Geko *et al.* [42] propose an ontology composed of 5 domains: Data, Organisation, Data Protection Principles, Data Subject Rights and Obligations, which serve as a knowledge base and is based on the idea of simplifying the understanding of GDPR in the context of information security. Indeed, the work consists in making an interrelation between GDPR and information security. The ontology has not been evaluated, and the authors do not provide access to the complete ontology or detailed documentation of the high-level ontology presented in the article. To simplify the task of developers in understanding and complying with GDPR, Pandit *et al.* [43] focus on the representation and modeling of personal data information in privacy policies in the form of an ontology. The ontology's concept definitions are based on two other ontologies, GDPR-tEXT & GDPRov. The aim of this ontology is to ensure a common representation of privacy policies and thus be used in multiple contexts. However, this work is limited to information relating to personal data and does not address certain aspects, such as user rights. In a similar context, Palmirani *et al.* [44] have developed an ontology divided into 5 modules (privacy agents, data types, processing operations, rights and obligations), which models the legal knowledge (GDPR). It thus offers greater coverage by integrating more concepts. The same ontology was taken up and refined in a more recent work [45].

Sangaroonsilp *et al.* [11] did not choose a formal representation. They use GDPR, ISO/IEC 29100, Thailand Personal Data Protection Act (Thailand PDPA) and Asia-Pacific Economic Cooperation (APEC) privacy framework to propose a taxonomy of privacy requirements. This taxonomy, comprising 71 requirements grouped into 7 categories, aims at developing privacy-friendly software applications. One of the uses of the taxonomy on which the authors base their work is the mapping (classification) of problem reports. The goal is to help organizations identify which requirements have not been met, and thus ensure compliance by taking these requirements into account.
In [46] the authors propose a method for ensuring GDPR compliance in the case of a personal data breach (PDB). It combines a structured data model based on GDPR legal terms and a business process model, enriched with simulations to assess and improve compliance. The methodology uses business process management (BPM) tools to automate compliance and optimize the use of technical and organizational resources. The proposed data model integrates key GDPR concepts, such as breach detection, mitigation measures, and notifications to authorities and data subjects. It structures and stores the information needed to effectively manage PDBs.
Bartolini *et al.* [47] view access control as essensial for GDPR compliance by protecting access to personal data. To achieve this, they propose a backlog, which consists of user stories that serve as the basis for defining the specific technical requirements needed to establish access control systems that comply with GDPR. Each user story is translated into an access control policy. Mougiakou *et al.* [48] propose another way of representing GDPR requirements. They use UML, in particular use case diagrams, to simplify the task of developers

and guide them in integrating privacy protection.

Our comparison table shows that some works focus solely on obtaining requirements and others on modeling the various aspects of GDPR without taking into account all key requirements of the regulation (partial coverage).
The proposed UML metamodels do not address all the aspects and principles of GDPR. For example [41], although it proposes an approach and a tool based on a metamodel, the latter is limited to the concepts found in privacy policies. The others present declarative models for the simplified reading of GDPR that are not fully exploitable in practice. In their part of the metamodel describing end-user rights, [39] present the concepts of data subject and rights, and show the relationships between them. However, no other attributes are added to make this metamodel exploitable (e.g., date of issuance of a right request or response from the application owner). This type of metamodels makes it easier to read the rules, but does not help in making operational the exercise of end-user rights.
Many of the works mentioned in this category have not had their findings verified by field experts. In addition, none of the aforementioned works offer comprehensive tools to help developers integrate GDPR requirements to proactively make their applications compliant.

Furthermore, by comparing our generic user stories to GDPR requirements proposed by Sangaroonsilp *et al.* [11] we have found that[22]: (1) Our user stories cover all identified requirements. (2) Our user stories offer broader coverage of GDPR. For instance, in our user stories we deal with the notion of minors and minimization that are not considered in the list of requirements.(3) Some of our user stories group several requirements of this related work. For example, the user story related to the right to erasure brings together several acceptance criteria, specifying the situations in which personal data must be erased (e.g., unlawful processing, objection, withdrawal of consent, etc.). In our work, this is a single user story with several acceptance criteria. In contrast, in [11], each acceptance criterion is formulated as a separate requirement. Thus, the user story conveys the user's overall need, while the requirements break down each scenario into more specific rules.

Compliance Checking. The other category of works aims to verify compliance with GDPR through its representation/elicitation of requirements. Krasnashchok *et al.* [49] proposed an ontology alignment mechanism covering the representation of GDPR articles relevant to privacy contract compliance checking. The main idea is the reuse and merging of ontological resources to allow the extraction of access control rules from contracts written in natural language. They chose a modeling based on Open Digital Rights Language for better flexibility and interoperability.

In Ribalta *et al.* [50] the authors have proposed an extension of the STS-ML (Security Requirements Modeling Tool) metamodel [51]. They added attributes to ensure compliance

[22]The results of this comparison reveal several significant points which are shared in the repository.

with GDPR principles. However, the proposed solution does not cover some important aspects of GDPR such as the right to portability and the right to objection. Torre *et al.* [52] propose a full model for checking compliance with GDPR, a glossary to make it easier to understand and compliance rules extracted from GDPR, encoded in OCL. The model is designed to serve as a basis for automated compliance verification methods but is not a solution for integrating privacy.

Other proposed works implemented solutions and proposed tools. Matulevičius *et al.* [53] extend Tom *et al.*'s proposal [39] and [54] with a tool that helps assessing compliance with GDPR standard. It compares the business processes of an application to reference GDPR processes using BPMN representations. In case of non-compliance, recommendations are proposed. However, this solution does not provide support for personal data policy management. Kirrane *et al.* [55] propose an OWL specific vocabulary to represent GDPR concepts. It includes a description of personal data, privacy policy and consents. This vocabulary is used in their platform for checking application compliance with GDPR at runtime thanks to the analysis of log files. Damiano Torre *et al.* [56] propose an approach to classify application privacy policies. The proposed model is based on deep learning techniques that detect metadata and thus check whether the content of a privacy policy is complete. The metadata were extracted from the regulation by the authors and presented in a conceptual model. The authors do not address all aspects and principles of GDPR such as breaches.

The authors in [57] propose a list of 45 requirements for *Data Processing Agreements* (DPA) compliance extracted from GDPR. This initial work is then exploited by the authors for the automatic verification of DPA compliance by proposing a solution based on NLP techniques. The aim is to see whether the sentences in the DPAs meet the requirements. The same authors [58] have used the previous work and [59] to propose a new approach to verifying the conformity of a DPA. In this work, the authors classify the content of a DPA according to the types of information in the conceptual model they have proposed. This new approach is based on ML methods instead of the NLP techniques presented in the previous work.

In contrast to our work, these works are downstream in the application lifecycle, and therefore do not proactively address the challenge of privacy integration. Detecting non-compliance implies corrective actions to resolve problems. Among these works, only one of them supports all the requirements of GDPR. Their instantiations can allow the quantification of the compliance rate of an application since the majority of attributes are of Boolean type. This makes them not adaptable to design solutions. Nevertheless, we emphasize the importance of these works, which we see as complementary to our approach.

Table 7: Related works comparison.

Work goal: *(i) Checking compliance (cc), (ii) Requirements modeling and elicitation (rm), GDPR Formal Representation (MOD), GDPR modeling coverage (MOD-C), Requirements (REQ), requirement coverage (REQ-C), Support (SUP), Experts intervention (EXP)*
P = Partially Covered, F = Fully Covered, ✗= Not applicable, ✓ applicable.

Privacy-based work	**Year**	**Work goal**	**MOD**	**MOD-C**	**REQ**	**REQ-C**	**SUP**	**EXP**
Diamantopoulou *et al.* [38]	2017	rm	UML Metamodel	P	✗	✗	✗	✗
Tom *et al.* [39]	2018	rm	UML Metamodel	P	✗	✗	✗	✗
Burneister *et al.* [4]	2019	rm	EA meta-model	P	✗	✗	✗	✗
Matulevičius *et al.* [53]	2020	cc	UML Metamodel	P	✗	✗	✓	✓
Kirrane *et al.* [55]	2020	cc	Ontology	P	✗	✗	✓	✓
Krasnashchok *et al.* [49]	2020	cc	ODRL Profil	P	✗	✗	✗	✗
Torre *et al.* [56]	2020	cc	hierarchical representation	P	✗	✗	✓	✓
Torre *et al.* [52]	2021	cc	UML model	F	✗	✗	✗	✓
Ribalta *et al.* [50]	2022	cc	UML Metamodel	P	✗	✗	✗	✗
Geko *et al.* [42]	2018	rm	Ontology	P	✗	✗	✗	✗
Mougiakou *et al.* [48]	2017	rm	✗	✗	UML use case	P	✗	✗
Pandit *et al.* [43]	2018	rm	Ontology	P	✗	✗	✗	✗
Palmirani *et al.* [45]	2018	rm	Ontology	P	✗	✗	✗	✓
Amaral *et al.* [57]	2023	cc	✗	✗	Listed	P	✓	✓
Sangaroonsilp *et al.* [11]	2023	rm	None	✗	Listed	P	✗	✗
Amaral *et al.* [58] [59]	2023	cc	hierarchical conceptual model	P	✗	✗	✓	✓
Bartolini *et al.* [47]	2019	rm	✗	✗	Listed	P	✗	✗
Caramujo *et al.* [41]	2019	rm	UML metamodel	P	✗	✗	✓	✗
Varela-Vaca *et al.* [46]	2025	rm	UML Metamodel	P	✗	✗	✓	✗
PRIAM	2025	rm	UML Metamodel	F	Generated	F	✓	✓

8. Conclusion and Future works

Privacy is becoming a major concern for application developers and owners, owing to technological advancements and the substantial increase in the collection and utilization of personal data. The objective is to protect users' privacy and ensure a high level of security. This awareness helps to enhance trust between end-users and application owners, who are custodians of their personal data. In this context, various standards and legal texts (including laws) have emerged to regulate the protection of user privacy. Among these, GDPR stands out as one of the strictest and most comprehensive regulations to date, imposing obligations on app owners to secure users' personal data.

This paper presents PRIAM, a metamodel that specifies concepts necessary for privacy requirement enforcement. PRIAM concepts support the identification of privacy-sensitive data and processing, along with the management of the associated users' rights as prescribed by GDPR. This metamodel has been evaluated and approved by experts in the field, complemented by an evaluation using LLMs. Our research work aims at proposing a tool-enabled software engineering method that makes it possible to developers to integrate personal data protection capabilities into existing software, thereby enhancing software privacy.

This method relies heavily on Model-Driven Engineering by leveraging PRIAM metamodel which is implemented as a DSL that is used for the generation of development artifacts: a backlog of user stories and a privacy enforcement database.

Our solution simplifies the work of developers by eliminating the need to read and understand the lengthy and complex text of GDPR, which includes legal-specific terms, or to seek assistance from experts. Furthermore, developers no longer have to manually translate GDPR into a conceptual model, saving time and reducing the risk of errors. As a generic solution, it can be integrated into any application. Another key aspect of our solution is that it does not depend on third-party hosting services. This allows companies to store and manage data according to their own policies. Moreover as it does not use personal data but only metadata, our solution is potentially compatible with any privacy preservation regulation.

Many perspectives remain open for future work. User stories and their related database are the first artifacts that we automatically generate from PRIAM models to assist the development process in the implementation of privacy requirements. They enable to estimate the extra development costs induced by privacy protection in applications. A short-term perspective is to support the full implementation of the user stories (limited so far to database extensions) thanks to generic microservices. This work will also involve evaluating PRIAM in real-world applications to quantify the effort developers must make to ensure GDPR compliance. The evaluation will focus specifically on PRIAM-DSL, whose automatically generated artifacts are used directly by microservices to enforce GDPR requirements. Furthermore, we could leverage the information in the PRIAM database to generate reports and conduct audits. Exploring this aspect would be beneficial for developing tools that simplify auditing processes.
The approach presented in this article, applied to the European regulation on the protection of personal data, could potentially be generalized to integrate any standard or regulation. This adaptability and extensibility would allow for the application of the approach to various regulatory frameworks.

The fact that the architecture that we propose is designed using the microservice style enables to some extent some flexibility. Based on the similarities between regulations (such as consent management or the right to rectification), our "modular" architecture can help in reusing a common base while enabling specific extensions to be added via specialized modules/microservices.
A detailed study of other regulations and laws would enable us to adapt the DSL and enhance the functionalities of existing microservices, or adding new regulation-specific microservices, to cover all possible requirements.
Finally, we plan to evaluate the scalability of our solution. Thanks to its modular and extensible architecture that uses the "database per service" pattern, we are quite confident that scalability can be guaranteed to some extent, and that we will be able to manage large

applications with complex data and many kinds of users on which multiple regulations should be respected.

9. Acknowledgments

Authors would like to warmly thank the anonymous respondents to the questionnaire.
This work was partially funded by the Eiffel Excellence Scholarship Program (2022–2023).

References

[1] R. Anderson, Security Engineering: A Guide to Building Dependable Distributed Systems, 3rd Edition, 1232 pages. Wiley, 2020.

[2] M. Palmirani, M. Martoni, A. Rossi, C. Bartolini, L. Robaldo, Pronto: Privacy ontology for legal reasoning, in: Electronic Government and the Information Systems Perspective: 7th International Conference, EGOVIS 2018, Regensburg, Germany, September 3–5, 2018, Proceedings 7, Springer, 2018, pp. 139–152.

[3] J.-H. Hoepman, Privacy design strategies, in: ICT Systems Security and Privacy Protection: 29th IFIP TC 11 International Conference, SEC 2014, Marrakech, Morocco, June 2-4, 2014. Proceedings 29, Springer, 2014, pp. 446–459.

[4] F. Burmeister, P. Drews, I. Schirmer, A privacy-driven enterprise architecture metamodel for supporting compliance with the general data protection regulation, Proc. of 52nd Hawaii International Conference on System Sciences (HICSS) 52 (2019) 6052–6061.

[5] E. Gómez-Martínez, M. Marroyo, S. T. Acuña, Towards the integration of the GDPR in the unified software development process, in: Proc. of 33rd International Conference on Software Engineering and Knowledge Engineering (SEKE), online, 2021.

[6] E. Grünewald, P. Wille, F. Pallas, M. C. Borges, M.-R. Ulbricht, TIRA: an OpenAPI extension and toolbox for GDPR transparency in RESTful architectures, in: 6th IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), IEEE, online, 2021, pp. 312–319.

[7] K. Fatema, E. Hadziselimovic, H. J. Pandit, C. Debruyne, D. Lewis, D. O'Sullivan, Compliance through informed consent: Semantic based consent permission and data management model., PrivOn@ISWC 1951 (2017) 1–16.

[8] D. Schmidt, Guest editor's introduction: Model-driven engineering, Computer 39 (2) (2006) 25–31. doi:10.1109/MC.2006.58.

[9] S. Lim, J. Oh, Navigating privacy: A global comparative analysis of data protection laws, IET Information Security 2025 (1) (2025) 5536763.

[10] K. Pohl, G. Böckle, F. van der Linden, Software Product Line Engineering - Foundations, Principles, and Techniques, Springer, 2005. doi:10.1007/3-540-28901-1.
URL https://doi.org/10.1007/3-540-28901-1

[11] P. Sangaroonsilp, H. K. Dam, M. Choetkiertikul, C. Ragkhitwetsagul, A. Ghose, A taxonomy for mining and classifying privacy requirements in issue reports, Information and Software Technology 157 (2023) 107162.

[12] S. Lamari, N. Benblidia, C. Tibermacine, C. Urtado, S. Vauttier, Leveraging a microservice architecture, access control and interoperability patterns to manage privacy-related user consents, in: W. Gaaloul, M. Sheng, Q. Yu, S. Yangui (Eds.), Service-Oriented Computing, Proc. 22nd International Conference on Service-Oriented Computing (ICSOC), Part II, LNCS 15405, Springer Nature, Tunis, Tunisia, 2025, pp. 146–157. doi:https://doi.org/10.1007/978-981-96-0808-9_12.

[13] General Data Protection Regulation (GDPR) compliance guidelines, https://gdpr.eu/, (Accessed on 04/04/2025) (2020).

[14] The open source pia software helps to carry out data protection impact assessment — cnil, https://www.cnil.fr/en/open-source-pia-software-helps-carry-out-data-protection-impact-assessment, (Accessed on 06/04/2025).

[15] MPS: The domain-specific language creator by jetbrains, https://www.jetbrains.com/mps/, (Accessed on 04/04/2025).

[16] Y. Wautelet, S. Heng, M. Kolp, I. Mirbel, Unifying and extending user story models, in: Advanced Information Systems Engineering: 26th International Conference, CAiSE 2014, Thessaloniki, Greece, June 16-20, 2014. Proceedings 26, Springer, 2014, pp. 211–225.

[17] Preparing your organisation for the general data protection regulation - your readiness checklist [online]. available at, https://www.dataprotection.ie/sites/default/files/uploads/2019-04/A-Guide-to-help-SMEs-Prepare-for-the-GDPR.pdf, (Accessed on 06/04/2025).

[18] Controllers checklist—ico [online]. available at, https://ico.org.uk/for-organisations/sme-web-hub/checklists/data-protection-self-assessment/controllers-checklist/, (Accessed on 06/04/2025).

[19] F. Pereira, P. Crocker, V. R. Leithardt, PADRES: Tool for Privacy, Data REgulation and Security, SoftwareX 17 (2022) 100895.

[20] GDPR checklist for data controllers [online]. available at, https://gdpr.eu/checklist/, (Accessed on 06/04/2025).

[21] EU GDPR readiness assessment tool [online]. available at, https://advisera.com/tools/eu-gdpr-readiness-assessment-tool/, (Accessed on 06/04/2025).

[22] Class diagram syntax and features, [Online; accessed 25/03/2025].
URL https://plantuml.com/class-diagram

[23] G. Marvin, N. Hellen, D. Jjingo, J. Nakatumba-Nabende, Prompt engineering in large language models, in: International conference on data intelligence and cognitive informatics, Springer, 2023, pp. 387–402.

[24] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, Q. V. Le, Finetuned language models are zero-shot learners, arXiv preprint arXiv:2109.01652 (2021).

[25] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al., Chain-of-thought prompting elicits reasoning in large language models, Advances in neural information processing systems 35 (2022) 24824–24837.

[26] Mistral AI, Mistral AI — frontier AI in your hands, https://mistral.ai/, (Accessed on 25/02/2025).

[27] A. Toumia, S. Szoniecky, I. Saleh, ColPri: Towards a collaborative privacy knowledge management ontology for the Internet of Things, in: Fifth International Conference on Fog and Mobile Edge Computing (FMEC), IEEE, Paris, France, 2020, pp. 150–157. doi:10.1109/FMEC49853.2020.9144927.

[28] S. Pape, K. Rannenberg, Applying privacy patterns to the internet of things' (IoT) architecture, Mobile Networks and Applications 24 (3) (2019) 925–933.

[29] F. Zohra Benhamida, J. Navarro Martín, O. Gómez Carmona, D. Casado Mansilla, D. López de Ipiña, A. Zaballos Diego, et al., PyFF: A fog-based flexible architecture for enabling privacy-by-design IoT-based communal smart environments, Sensors 21 (11) (2021).

[30] M. Gharib, P. Giorgini, J. Mylopoulos, Towards an ontology for privacy requirements via a systematic literature review, in: International conference on conceptual modeling, Springer, 2017, pp. 193–208.

[31] D. M. Nogueira, C. Maciel, J. Viterbo, D. Vecchiato, A privacy-driven data management model for smart personal assistants, in: International Conference on Human Aspects of Information Security, Privacy, and Trust, Springer, 2017, pp. 722–738.

[32] GDPR compliance cloud platform for micro enterprises (SMOOTH), h2020 european project, https://smoothplatform.eu/, (Accessed on 28/03/2025).

[33] Data governance framework for supporting GDPR (DefenD), H2020 European project, https://www.defendproject.eu/, (Accessed on 06/04/2025).

[34] Protection and control of secured information by means of a privacy enhanced dashboard (PoSeID-on), H2020 European project, https://www.poseidon-h2020.eu/, (Accessed on 28/03/2025).

[35] Platform for privacy preserving data analytics (PAPAYA), h2020 european project, https://www.papaya-project.eu/, (Accessed on 28/03/2025).

[36] Privacy and data protection 4 engineering (PDP4E), h2020 european project, https://www.pdp4e-project.eu/, (Accessed on 06/04/2025).

[37] Business process re-engineering and functional toolkit for GDPR compliance (BPR4GDPR), https://www.bpr4gdpr.eu/, (Accessed on 06/04/2025).

[38] V. Diamantopoulou, K. Angelopoulos, M. Pavlidis, H. Mouratidis, A metamodel for GDPR-based privacy level agreements., in: ER Forum/Demos, Vol. 1979, 2017, pp. 285–291.

[39] J. Tom, E. Sing, R. Matulevičius, Conceptual representation of the GDPR: model and application directions, in: International Conference on Business Informatics Research, Springer, 2018, pp. 18–28.

[40] The TOGAF Standard, https://pubs.opengroup.org/architecture/togaf9-doc/arch/, (Version 9.2, Accessed on 04/04/2025).

[41] J. Caramujo, A. Rodrigues da Silva, S. Monfared, A. Ribeiro, P. Calado, T. Breaux, RSL-IL4Privacy: A domain-specific language for the rigorous specification of privacy policies, Requirements Engineering 24 (2019) 1–26. doi:https://doi.org/10.1007/s00766-018-0305-2.

[42] M. Geko, S. Tjoa, An ontology capturing the interdependence of the general data protection regulation (GDPR) and information security, in: Proc. of the Central European Cybersecurity Conference (CECC), ACM, Ljubljana, Slovenia, 2018, pp. 1–6.

[43] H. J. Pandit, D. O'Sullivan, D. Lewis, An ontology design pattern for describing personal data in privacy policies., in: WOP@ ISWC, 2018, pp. 29–39.

[44] M. Palmirani, M. Martoni, A. Rossi, C. Bartolini, L. Robaldo, Pronto: Privacy ontology for legal reasoning, in: Electronic Government and the Information Systems Perspective: 7th International Conference, EGOVIS 2018, Regensburg, Germany, September 3–5, 2018, Proceedings 7, Springer, 2018, pp. 139–152.

[45] M. Palmirani, G. Bincoletto, V. Leone, S. Sapienza, F. Sovrano, Hybrid refining approach of pronto ontology, in: International Conference on Electronic Government and the Information Systems Perspective, Springer, 2020, pp. 3–17.

[46] Á. J. Varela-Vaca, M. T. Gómez-López, Y. Morales Zamora, R. M Gasca, Business process models and simulation to enable GDPR compliance, International Journal of Information Security 24 (1) (2025) 1–21.

[47] C. Bartolini, S. Daoudagh, G. Lenzini, E. Marchetti, GDPR-based user stories in the access control perspective, in: International Conference on the Quality of Information and Communications Technology, Springer, 2019, pp. 3–17.

[48] E. Mougiakou, M. Virvou, Based on GDPR privacy in UML: Case of e-learning program, in: 2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA), IEEE, 2017, pp. 1–8.

[49] K. Krasnashchok, M. Mustapha, A. Al Bassit, S. Skhiri, Towards privacy policy conceptual modeling, in: International Conference on Conceptual Modeling, Springer, 2020, pp. 429–438.

[50] C. Negri-Ribalta, R. Noel, N. Herbaut, O. Pastor, C. Salinesi, Socio-technical modelling for GDPR principles: an extension for the STS-ml, in: IEEE 30th International Requirements Engineering Conference Workshops (REW), IEEE, Melbourne, Australia, 2022, pp. 238–234. doi:10.1109/REW56159.2022.00052.

[51] M. Robol, M. Salnitri, P. Giorgini, Toward GDPR-compliant socio-technical systems: modeling language and reasoning framework, in: 10th IFIP WG 8.1. Working Conference on the Practice of Enterprise Modeling (PoEM), Proceedings 10, Springer, Leuven, Belgium, 2017, pp. 236–250.

[52] D. Torre, M. Alferez, G. Soltana, M. Sabetzadeh, L. Briand, Modeling data protection and privacy: application and experience with GDPR, Software and Systems Modeling 20 (6) (2021) 2071–2087.

[53] R. Matulevičius, J. Tom, K. Kala, E. Sing, A method for managing GDPR compliance in business processes, in: Advanced Information Systems Engineering: 32nd International Conference,CAiSE Forum 2020, Grenoble, France, June 8–12, 2020, Proceedings, Springer, 2020, pp. 100–112. doi:10.1007/978-3-030-58135-0.

[54] K. Kala, Refinement of the general data protection regulation (gdpr) model: administrative fines perspective, Ph.D. thesis, Master's thesis, University of Tartu (2019).

[55] S. Kirrane, J. D. Fernández, P. Bonatti, U. Milosevic, A. Polleres, R. Wenning, The special-k personal data processing transparency and compliance platform, arXiv preprint arXiv:2001.09461 (2020).

[56] D. Torre, S. Abualhaija, M. Sabetzadeh, L. Briand, K. Baetens, P. Goes, S. Forastier, An ai-assisted approach for checking the completeness of privacy policies against GDPR, in: IEEE 28th International Requirements Engineering Conference (RE), IEEE, 2020, pp. 136–146.

[57] O. A. Cejas, M. I. Azeem, S. Abualhaija, L. C. Briand, NLP-based automated compliance checking of data processing agreements against GDPR, IEEE Transactions on Software Engineering 49 (9) (2023) 4282–4303. doi:10.1109/TSE.2023.3288901.

[58] O. Amaral, S. Abualhaija, L. Briand, ML-based compliance verification of data processing agreements against GDPR, in: IEEE 31st International Requirements Engineering Conference (RE), IEEE, Hannover, Germany, 2023, pp. 53–64. doi:10.1109/RE57278.2023.00015.

[59] O. Amaral, S. Abualhaija, M. Sabetzadeh, L. Briand, A model-based conceptualization of requirements for compliance checking of data processing against GDPR, in: IEEE 29th International Requirements Engineering Conference Workshops (REW), IEEE, Notre Dame, USA, 2021, pp. 16–20. doi:10.1109/REW53955.2021.00009.