

A UML Profile for Privacy-Aware Data Lifecycle Models

Majed Alshammari and Andrew Simpson

Department of Computer Science, University of Oxford
Wolfson Building, Parks Road, Oxford OX1 3QD, UK
`firstname.secondname@cs.ox.ac.uk`

Abstract. Concerns over data-processing activities that may lead to privacy violations or harms have motivated the development of legal frameworks and standards to govern the processing of personal data. However, it is widely recognised that there is a disconnect between policy-makers’ intentions and software engineering reality. The Abstract Personal Data Lifecycle (APDL) model, which was proposed to serve as an abstract model for personal data life-cycles, distinguishes between the main operations that can be performed on personal data during its life-cycle by outlining the various distinct activities for each operation. We show how the APDL can be represented in terms of the Unified Modeling Language (UML). The profile is illustrated via a realistic case study.

1 Introduction

Privacy concerns have motivated the development of legal frameworks and standards for governing the processing of personal data, such as the EU General Data Protection Regulation (GDPR) [23] and the Global Privacy Standard (GPS) [5]. Typically, legal frameworks and standards are given at a high level of abstraction without relying on rigorous models that explicitly specify privacy-related concepts and associated properties [3]. In parallel, Privacy by Design (PbD) [6] has been advocated as a proactive and integrative approach for embedding privacy into the early stages of the design process. However, the principles of PbD are given at a high level of abstraction, which, in turn, leads to challenges with regards to translating these principles into engineering activities and artefacts [8].

One attempt to address these challenges and bridge the gap between policy-makers and software engineers is a set of activities for privacy engineering [8]. However, these are not accompanied by guidelines that aid engineers in analysing functional requirements or identifying potential privacy risks in a structured and contextual manner. As another example, the privacy design strategies of [9] jump directly from abstract legal principles into software architecture design [13] without providing criteria for making architectural choices or for justifying these choices with respect to a privacy risk analysis [10].

To realise PbD and translate its principles into engineering activities, a systematic methodology for Privacy Impact Assessments (PIAs) is required [16]. A

PIA is defined as a process that identifies and mitigates the impact of an initiative on privacy with multiple stakeholders' participation [16, 25]. Accordingly, the first step of a PIA is to describe data-processing activities so that potential privacy risks can be analysed and assessed [16]. Additionally, to support a meaningful participation of multiple stakeholders, a common language needs to be provided to facilitate communication between those stakeholders. As such, the first step towards bridging the gap between policy-makers and software engineers involves providing a data management model built upon an appropriate conceptual model for privacy engineering. Such a model serves as a stepping stone for modelling privacy-related concepts along with associated properties and relationships, and for representing data-processing activities in a way that is amenable to analysis. The Abstract Personal Data Lifecycle (APDL) model [1] was developed to achieve these goals. It represents the processing of personal data in terms of states (data items), operations (data-processing activities), and roles (actors). It facilitates understanding of the meaning of privacy-related concepts and the ways in which systems can be developed to comply with legal frameworks and standards, and to meet data subjects' expectations by supporting the traceability and management of the flow of personal data. The APDL model [1], however, was informally represented in terms of the lifecycle stages, associated activities and involved actors. In order for it to be integrated into an appropriate software engineering process, a widely-used modelling notation needs to be adopted to help support its main concepts. The Unified Modeling Language (UML) [15] is ideal for this purpose.

UMLsec [11] was introduced as a profile to represent security-related concepts using the standard extension mechanisms of the UML meta-model, i.e. stereotypes, tagged values and constraints. Stereotypes are used, along with tagged values to represent the key aspects of security as requirements and assumptions, and constraints associated to stereotypes give criteria for the evaluation of these aspects in a given specification to determine whether the requirements are satisfied [12]. We use the same extension mechanisms to represent the privacy-related concepts introduced by the APDL model. Specifically, we present a conceptual model, upon which we define a profile that allows the APDL model to be represented in the UML as a meta-model. The meta-model represents the personal data lifecycle in terms of stages that involve data-processing activities that consist of concrete actions and corresponding events, and roles that define a set of responsibilities performed by different actors according to their capabilities.

2 The Abstract Personal Data Lifecycle Model

We start by providing an overview of the Abstract Personal Data Lifecycle (APDL) and by stating a set of principles for a conceptual model.

2.1 An Overview of the APDL model

The APDL model [1] was proposed to serve as an abstract model for personal data lifecycles — where a data lifecycle is defined by a set of stages through

which personal data moves during its lifetime, associated activities, and involved actors. It aims to specify and represent the minimum amount of personal data, along with possible data-processing activities that are necessary for the specified purpose.

The APDL model distinguishes between the main types of operations that can be performed on personal data during its lifetime. For each operation, it outlines distinct activities that can be conducted in an ordered and planned manner. The model categorises these activities into the following lifecycle stages: initiation, collection, retention, access, review, usage, disclosure and destruction. Each stage involves a set of activities that may be performed by one or more types of role: data modellers, data subjects, data controllers, data processors and third parties. Each role specifies a set of related activities that are expected to be performed together by different actors according to their capabilities and responsibilities. Such a classification reflects the extent to which the flow of personal data is appropriate in terms of involved actors and their assigned roles and responsibilities. Each lifecycle stage is restricted by a set of the GPS principles [5] to govern the behaviour of associated activities. For example, ‘purposes’ emphasises that the purposes for which personal data is collected, retained, used and disclosed must be clearly specified and communicated to data subjects at or before the time of collection, and ‘collection limitation’ affirms that the collection of personal data must be fair, lawful and limited to the specified purposes.

2.2 Principles

In order to develop an appropriate conceptual model, essential principles for the core parts of the model, which will be used as the basis for our UML profile, need to be specified. Crucially, the *purpose* and *scope* of modelling shall be specified in relation to the context of data protection. In addition, the *most appropriate technique* shall be used for deriving useful and potentially usable concepts, associated properties and relationships.

We partially specify the purpose and scope of the modelling, as well as the appropriate techniques and conceptualisation approach used.

1. The purpose of building a conceptual model is to describe precisely the key privacy-related concepts, associated properties and relationships in the context of data protection. The model is intended to be used by multiple stakeholders — both those concerned with data protection and those responsible for developing and maintaining privacy-preserving systems. The model is intended to be used as a common language for privacy engineering to consider protection, manageability and traceability of personal data. Such a language is provided with the ability to express stakeholders’ expectations and concerns.
2. The scope of the modelling is identified by a list of concepts (as explained in Section 3).
3. Informal text analysis is used with the aim of analysing commonly-used concepts that have been already described in privacy standards instead of

‘starting from scratch’. We have chosen the Global Privacy Standard (GPS) principles [5] and the Generally Accepted Privacy Principles (GAPP) [2] to ensure that our modelling of the privacy-related concepts and their meanings are based upon a widely-used set of terms. These standards are based on internationally known Fair Information Practice Principles (FIPPs) [24].

4. Concept classification is used with the aim of analysing and classifying relevant terms into concepts and processing activities that can be represented in a fine-grained manner as actions. With regards to concepts, we identify and describe useful and potentially usable concepts, associated properties, meanings and possible values. With regards to actions, we identify and describe useful and potentially usable actions and associated constraints that specify conditions to be satisfied before, or to be guaranteed after, the execution of corresponding actions.

3 A Conceptual Model for the APDL

We now define a conceptual model for the APDL.

3.1 Purpose and personal data

Purpose represents goals and reasons for which personal data is collected and processed. It has the following properties: *informalDescription*; *actualPurpose* (in a concrete and explicit manner); *isFair* (indicates whether the processing of personal data has justified adverse effects on the concerned data subjects and is consistent with their reasonable expectations); *isLawful* (indicates whether there are legitimate or legal grounds for collecting and processing personal data); *isProportional* (indicates whether the specified purpose is legally and politically assessed in terms of proportionality); and *relevantPrinciple* (specifies the relevant GPS principles that govern the purpose specification in the sense of placing limitations or constraints).

For a purpose to be fulfilled, a minimum amount of personal data needs to be appropriately specified. As such, **PersonalData** represents the minimum necessary amount of data that is sufficiently related to an identified or identifiable individual in support of the specified purpose. It has the following properties: *informalDescription*; *category* (indicates the category of personal data in terms of its sensitivity and the manner in which it is to be processed, and drawn from {SpecialCategory, Unspecified}¹); and *type* (indicates the type of personal data in relation to the source and manner in which it is created, and drawn from {Collected, Acquired, Derived}).

In order to specify the required data that fulfils the specified purpose, a data model needs to be constructed; **DataModelling** represents the relevant objects, associated properties, relationships and constraints for the purpose of specifying the required data. This representation can be used as shared knowledge by

¹ Personal data, by its nature, is considered sensitive data when it is related to special categories, including racial or ethnic origin, etc. [23].

multiple stakeholders for a specific application. It has the following properties: *subjectDomain*; *modellingPurpose*; *modellingScope* (specifies the set of objects to be represented at an appropriate level of abstraction); and *dataModel* (refers to the model that represents the relevant objects, their properties and relationships to be used as shared knowledge in the domain of interest).

3.2 Lifecycle and associated stages

DataLifecycle represents the main characteristics of the personal data lifecycle in terms of the openness of the processed data and the centrality of its underlying system. It has the following properties: *informalDescription*; *isOpen*; and *isCentralised*. Each data lifecycle consists of a set of stages. As such, **LifecycleStage** represents the concept of a generic lifecycle stage that models all possible stages through which personal data moves during its lifecycle in more repetitive and circular flows. The LifecycleStage is abstractly represented as a general classifier that can be used as a classification of all possible stages of the lifecycle. It is mainly used as a target of generalisations, which can be specialised into eight specific classifiers according to associated activities. We consider each in turn.

Initiation represents a complete processing plan that can be referred to before and during the processing of personal data. This plan is a prerequisite for establishing and representing a privacy notice. It has the following properties: *informalDescription*; *specifiedPurpose*; *requiredData* (specifies the minimum amount of personal data as a set of relevant, adequate and not excessive personal data items in support of the specified purpose); *dataSource* (specifies the sources from which personal data items are to be collected, derived or acquired, whether these are internal or external sources); *availableChoice* (describes the choices available to data subjects with regards to the collection, usage and disclosure of their personal data); *consentType* (indicates the type of consent that needs to be obtained in relation to the degree of personal data sensitivity and the manner in which is to be processed, with values drawn from {Explicit, Implicit}); *collectionMethod*; *storageMethod*; *retentionTime* (specifies the necessary period for which personal data is retained to fulfil the specified purpose or as required by applicable laws and regulations); *retrievalMechanism* (specifies the means and the manner in which personal data is to be retrieved or consulted, including query languages, command-line, browser, or graphical user interfaces); *disclosureMechanism* (specifies the means and the manner in which personal data is to be disseminated, transmitted or made available); *destructionMethod*; *applicableRegulation* (indicates the applicable laws and regulations); and *relevantPrinciple* (specifies the relevant GPS principles that govern the stage of lifecycle in the sense of placing limitations or constraints on the associated activities).

Collection represents the act of creating personal data values, whether these are directly recorded or collected from data subjects, or have been acquired from external sources. It has the properties *informalDescription*, *createdData*, *dataSource*, *collectionMethod*, *availableChoice*, *consentType*, and *relevantPrinciple*.

Retention represents the organising, structuring or storing of personal data values in repositories or digital storage media for operational, compliance or op-

erational recovery purposes. It has the properties *informalDescription*, *retainedData*, *activityType* (drawn from {PrimaryStorage, Archiving, Backup}), *activityPurpose* (drawn from {Operational, RegulatoryCompliance, FutureReference, OperationalRecovery}), *retentionTime*, *storageMethod*, and *relevantPrinciple*.

Access represents the act of specifying, and retrieving or consulting personal data values that are stored in repositories or digital storage media. The aim is to make this data accessible and ready for use in relation to the specified purpose. It has the following properties: *informalDescription*; *retrievedData*; *retrievalMechanism*; and *relevantPrinciple*.

Review represents the act of implementing the access right and rectifying personal data values by data subjects to ensure that their data is accurate, complete and up-to-date. It has the following properties: *informalDescription*; *reviewedData*; *activityType* (drawn from {Adaptation, Alteration, Alignment}); *activityPurpose* (drawn from {Update, Correction}); and *relevantPrinciple*.

Disclosure represents the act of disseminating, making available or transmitting personal data for external use by third parties. It has the following properties: *informalDescription*; *disclosedData*; *activityType* (drawn from {InitialProcessing, FurtherProcessing}); *activityPurpose* (drawn from {TheSpecifiedPurposes, HistoricalPurposes, ScientificPurposes, StatisticalPurposes}); *disclosureMechanism*; and *relevantPrinciple*.

Usage represents the act of using, altering, adapting, refining, aligning or combining personal data items. It has the properties *informalDescription*, *usedData*, *activityType* (drawn from {Adaptation, Alteration, Alignment, Combination}), *activityPurpose* (drawn from {Use, Derivation}), and *relevantPrinciple*.

Destruction represents the act of erasing, destroying, redacting or disposing of personal data. It has the following properties: *informalDescription*; *destroyedData*; *destructionMethod*; *competentAuthority* (drawn from {InternalUnit, ExternalDataDestructionService}); and *relevantPrinciple*.

3.3 Stage activities, events and actions

StageActivity represents data-processing activities that constitute the operations performing on personal data in each stage of the lifecycle. It has the following properties: *informalDescription*; *input*; *output*; *preCondition*; and *postCondition*.

StageEvent represents occurrences that may happen at specific points in time that may have consequences for personal data. It has two properties: *informalDescription* and *category* (*implicit* events occur on the change of states or the passage of some interval of time; *explicit* events occur when an operation is directly requested).

StageAction represents single execution steps within an activity. Actions are the fundamental units that describe personal data processing activities in a fine-grained manner. It has the following properties: *informalDescription*; *inputParameter*; *outputParameter*; *localPreCondition*; and *localPostCondition*.

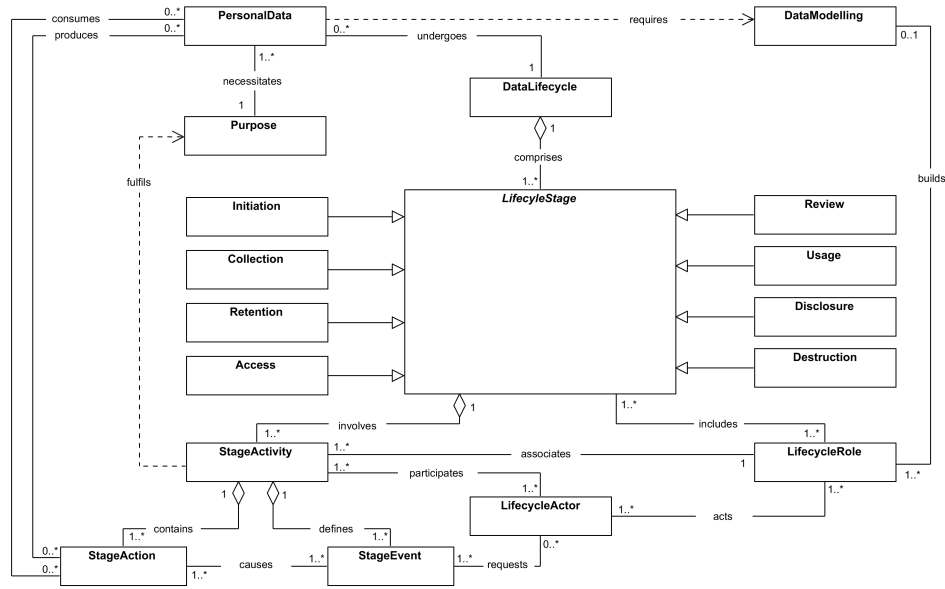


Fig. 1. The meta-model of the APDL profile.

3.4 Lifecycle roles and actors

LifecycleRole represents the way in which a concerned actor participates in a set of related activities of the personal data lifecycle. As such, a role represents a set of responsibilities that are logically related to each other, either by their objectives or by the actors that may play the role. It has the following properties: *informalDescription*; *roleType* (drawn from {DataModeller, DataSubject, DataController, DataProcessor, ThirdParty}); and *responsibility*. Lifecycle roles are identified to cover all stages of the data lifecycle. The processing of personal data in various stages identifies actors in these roles.

LifecycleActor represents an external or internal entity that is capable of, and responsible for, performing the activities of the role to which is assigned. It has the following properties: *informalDescription*; *actorNature* (drawn from {HumanActor, SoftwareAgent}); and *responsibility*.

Figure 1 shows the meta-model of the APDL profile with minimal syntax, omitting the attributes of the classes for simplicity and readability.

4 A UML Profile for the APDL Model

We now map the conceptual model of Section 3 to the UML profile. The stereotypes of the APDL profile are defined to extend existing metaclasses with the aim of using privacy-related terminology whether in place of, or in addition to, the terminology used for the extended metaclasses. The abstract syntax of the

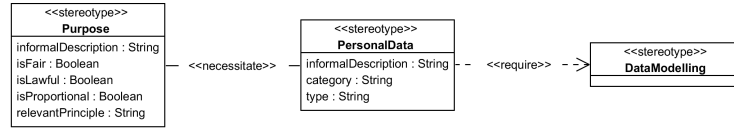


Fig. 2. Purpose, PersonalData and DataModelling stereotypes at the meta-level.

APDL is specified by extending three elements of the UML metamodel — the metaclass *Class*, the metaclass *Association* and the metaclass *Dependency* — with additional properties and constraints. The name of the applied stereotypes are shown within a pair of guillemets. The UML profile defines the concepts needed to model personal data processing activities using UML 2.5 [15]. The constraints needed to express privacy-related concepts of the APDL model are limited to association multiplicities, pre- and post-conditions of stage activities and actions unless additional constraints are explicitly stated.

We distinguish between three levels of abstraction: meta, domain and instance levels. Due to space restrictions, we focus on meta and domain levels only. At the meta-level, we refer to domain-independent abstractions, such as stereotypes, tag definitions, relationships and constraints in relation to the conceptual model. At the domain level, i.e. when a stereotype is applied to a model element, we refer to stereotyped classes, tagged values, domain-specific attributes, relationships and constraints specific to the application domain.

The stereotyped classes and associations belonging to the APDL profile are listed in Table 1.

4.1 Purpose, PersonalData and DataModelling

Stereotypes. The purpose for which personal data is processed can be specified using the «Purpose» stereotype, which constrains the semantics of the objects, meaning that only they can be used as purposes. The primary tag definitions of the Purpose stereotype are *informalDescription*, *isFair*, *isLawful*, *isProportional* and *relevantPrinciple*, as illustrated in Figure 2. When the Purpose stereotype is applied to any class, its primary attributes may include *actualPurpose*. Some aspects are more challenging to model, such as the fairness, lawfulness and proportionality of the specified purpose. These can be represented as *Boolean* tagged values to be specified by competent or authorised actors. The primary tag definitions of the «PersonalData» stereotype are *informalDescription*, *category* and *type*, as illustrated in Figure 2. When the PersonalData stereotype is applied to any class, its primary attributes are specific to the application domain.

Relationships. «Necessitate» is an association that denotes a relationship between «Purpose» and «PersonalData». It is used to specify a minimum amount of data required to fulfil the specified purpose, as illustrated in Figure 2. «Require» is a usage dependency that denotes a relationship between «RequiredData» and «DataModelling». It specifies that the required data that fulfils the specified purpose requires a data model for its full specification, as per Figure 2.

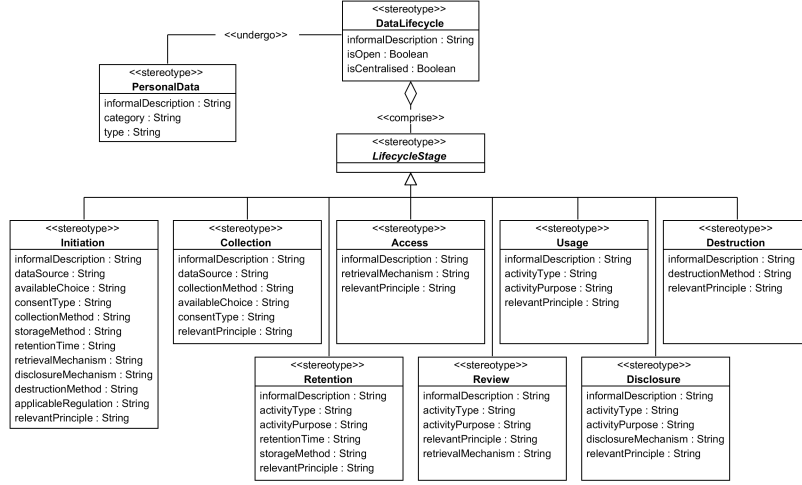


Fig. 3. DataLifecycle and LifecycleStage stereotypes at the meta-level.

Constraints. Each data-processing initiative has an abstract purpose that can be concretely refined to a set of purposes that can be specified in terms of data-processing activities, which, in turn, can be hierarchically structured as concrete actions and events. Consequently, it should not make sense to have multiple instances of the class stereotyped by «Purpose». As such, it is constrained as a singleton — it is instantiated only once in each particular model.

4.2 DataLifecycle, LifecycleStage and its specialisations

Stereotypes. The stages through which personal data moves during its lifetime can be specified using the «DataLifecycle» stereotype. The primary tag definition of the «DataLifecycle» stereotype is: *informalDescription*, *isOpen* and *isCentralised*, as illustrated in Figure 3. When the DataLifecycle stereotype is applied to any class, its primary attributes are specific to the application domain. The DataLifecycle stereotype consists of one or more stages that involve data processing activities. These stages are represented by the abstract «LifecycleStage» stereotype. The primary tag definitions of its specialisations are illustrated in Figure 3.

Relationships. «Undergo» is an association that denotes a relationship between «PersonalData» and «DataLifecycle». The undergo association is used to specify that personal data is subject to a set of stages, each of which involves a set of processing activities. «Comprise» is a binary association that denotes a relationship between «DataLifecycle» and «LifecycleStage». The comprise association is used to specify that the data lifecycle consists of various stages that involve distinct but related processing activities.

Constraints. «LifecycleStage» represents the concept of a generic lifecycle stage. It is mainly used as a target of generalisation; as such, it is constrained as

abstract. The generalisation set, which combines all the special classifiers of the LifecycleStage, has two properties: *complete* and *disjoint*. Each data-processing initiative requires the development of a complete processing plan that may serve as the basis of establishing a privacy notice to be communicated to data subjects. Consequently, it should not make sense to have multiple instances of the «Initiation», which represent the processing plan. As such, it is constrained as a singleton.

4.3 StageActivity, StageEvent and StageAction

Each lifecycle stage involves a set of activities, which, in turn, consist of a set of concrete actions, as well as a set of events that trigger their execution.

Stereotypes. A data-processing activity is specified using the «StageActivity» stereotype, which constrains the semantics of the objects, meaning that only they can be used as processing activities. The primary tag definitions of the StageActivity stereotype are *informalDescription*, *preCondition* and *postCondition*, as illustrated in Figure 4. When the StageActivity stereotype is applied to any class its primary attributes may include *input* and *output*. Each StageActivity contains a set of actions specified by the «StageAction» stereotype. The primary tag definitions of the StageAction stereotype are *informalDescription*, *localPreCondition* and *localPostCondition*, as illustrated in Figure 4. When the StageAction stereotype is applied to any class, its primary attributes may include *inputParameter* and *outputParameter*. Each StageActivity defines a set of events specified using the «StageEvent» stereotype. The primary tag definitions of the StageEvent stereotype are *informalDescription* and *category*, as illustrated in Figure 4. When the StageEvent stereotype is applied to any class, its primary attributes are specific to the application domain.

Relationships. Figure 4 shows how LifecycleStage, StageActivity, StageAction and StageEvent stereotypes participate in relationships.

«Involve» is a binary association that denotes a relationship between «LifecycleStage» and «StageActivity». Each lifecycle stage may involve one or more related but distinct activities. «Contain» is a binary association that denotes a relationship between «StageActivity» and «StageAction». Each stage activity may consist of one or more concrete actions. «Define» is a binary association that denotes a relationship between «StageActivity» and «StageEvent». Each stage activity may consist of one or more events that are requested by actors to trigger its execution. «Cause» is an association that denotes a relationship between «StageEvent» and «StageAction». Each action may be triggered by zero or more events that are requested by actors. «Consume» is an association that denotes a relationship between «StageAction» and «PersonalData». The action requires specific personal data items to accomplish its execution. «Produce» is an association that denotes a relationship between a «StageAction» class and a «PersonalData» class. The action provides specific personal data items as a result of its execution. «Fulfil» is an abstraction that denotes a refinement relationship between «StageActivity» and «Purpose». It is a specialisation of the standard abstraction stereotype «Refine». It is used to specify data-processing

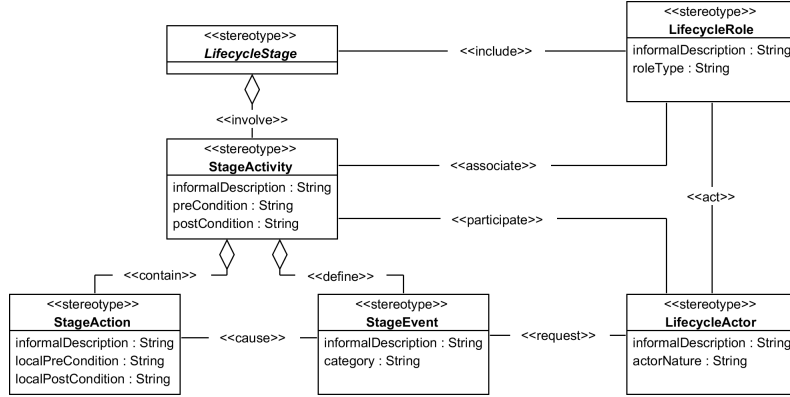


Fig. 4. StageActivity, StageAction, StageEvent, LifecycleRole and LifecycleActor stereotypes at the meta-level.

activities that have already been specified at a certain level of detail as a purpose. It is used to capture how a processing activity participates in the fulfilment of the specified purpose.

Constraints. The constraint of the «Consume» stereotype is that values of the association's *inputParameter* property must be attributes of the «Personal-Data» stereotyped class. Similarly, the constraint of the «Produce» stereotype is that values of the association's *outputParameter* property must be attributes of the «PersonalData» stereotyped class.

4.4 LifecycleRole and LifecycleActor

Each lifecycle stage involves a set of roles that may be played by different actors.

Stereotypes. A set of related activities that are expected to be performed together can be represented using the «LifecycleRole» stereotype. The primary tag definitions of the LifecycleRole stereotype are *informalDescription* and *roleType*, as illustrated in Figure 4. When the LifecycleRole stereotype is applied to any class, its primary attributes are specific to the application domain. An external or internal entity that is capable of, and responsible for, performing a set of activities associated with the role to which is assigned can be specified using the «LifecycleActor» stereotype. The primary tag definitions of the LifecycleActor stereotype are *informalDescription* and *actorNature*, as illustrated in Figure 4. When the LifecycleActor stereotype is applied to any class, its primary attributes are specific to the application domain.

Relationships. Figure 4 shows how the LifecycleRole and LifecycleActor stereotypes participate in relationships.

«Include» is an association that denotes a relationship between «LifecycleStage» and «LifecycleRole». Each lifecycle stage may include one or more lifecycle roles that participate to accomplish associated activities, and each lifecycle role may participate in one or more lifecycle stages. «Associate» is an

association that denotes a relationship between «StageActivity» and «LifecycleRole». Each stage activity may be assigned to exactly one lifecycle role; further, each lifecycle role may involve one or more stage activities. «Act» is an association that denotes a relationship between «LifecycleActor» and «LifecycleRole». Each lifecycle actor may be assigned to one or more lifecycle roles; further, each lifecycle role may involve one or more lifecycle actors. «Request» is an association that denotes a relationship between «LifecycleActor» and «StageEvent». Each lifecycle actor may perform an action by requesting one or more stage events that trigger its execution, and each stage event may be requested by zero or more lifecycle actor. «Participate» is an association that denotes a relationship between «LifecycleActor» and «StageActivity». Each lifecycle actor is capable of performing one or more stage activities that are assigned to one or more lifecycle roles to which the actor is assigned, and each stage activity may involve one or more lifecycle actors. «Build» is an association that denotes a relationship between «LifecycleRole» and «DataModelling». Each data modeller may construct zero or one data models, and each data model may be constructed by one or more data modellers.

5 The European Electronic Toll Service

5.1 Overview

The aim of the European Electronic Toll Service (EETS) [18] is to support interoperability between electronic road toll systems. We have chosen this case study for the following reasons. First, it has been critically analysed with regards to privacy concerns in the literature [8, 4]. Second, EETS is regulated by Directive 2004/52/EC of the European Parliament and of the Council of 29 April 2004 on the interoperability of electronic road toll systems in the Community [21] and the related Commission Decision 2009/750/EC of 6 October 2009 on the definition of the European Electronic Toll Service and its technical elements [22]. Third, the European Commission provides full details about EETS by publishing a guide as a reference manual for all parties concerned by the Directive and the Decision. The guide illustrates references and procedures to help the implementation of electronic road toll systems interoperability and EETS [18].

EETS complements national electronic road toll systems to ensure their interoperability. It is intended to cover all domains and objects that are subject to toll, such as road networks, specific sections of roads (e.g. a bridge, a tunnel or a ferry connection), or specific areas offering services (e.g. a parking lot or access to a protected area in a city). It enables road users to easily pay road-usage tolls throughout the Member States with a single subscription contract with a service provider [18].

The main parties involved in the EETS are users, service providers and toll chargers. Service providers are legal entities that grant access to EETS to road users [22]. Toll chargers are public or private organisations that are responsible for levying tolls for the circulation of vehicles in an EETS domain [22]. A user is a natural or legal person who subscribes to a service provider in order to get

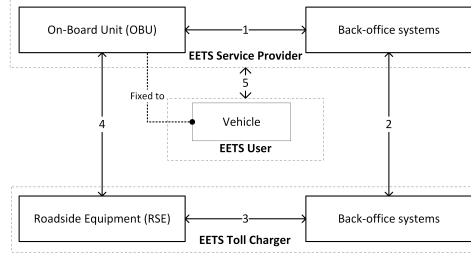


Fig. 5. The essential elements of the EETS architecture.

access to EETS, regardless of nationality, country of residence or the Member State in which the vehicle is registered [22]. By signing a contract, a user needs to provide a set of data — user and vehicle classification parameters — specified by a responsible toll charger, as well as to be informed about the processing of their personal data in relation to applicable law and regulations. Accordingly, the service provider provides the user with an On-Board Unit (OBU) to be installed on-board a vehicle to collect, store, and remotely receive and transmit time, distance and location data over time. This data, together with the user’s and vehicle’s parameters, are specified to declare the toll of circulating a vehicle in a specific toll domain [18].

Prior to explaining the interfaces, we introduce some relevant concepts [22]. *Toll domain* is a road network or a specific section of a road that is subject to toll. *Toll declaration* is a statement sent to a toll charger to confirm the circulation of a vehicle in its a toll domain. *Toll transaction* is an action or a set of actions in which a toll declaration is sent to a toll charger. *Toll context data* is data defined by the responsible toll charger to describe the location of the toll domain, charging policies, and the format of toll declarations. *Domain statement* is a statement that is developed and maintained by a toll charger to establish a set of conditions for a service provider for accessing its toll domains.

Figure 5 illustrates the essential elements of the EETS architecture, together with interfaces via which data is exchanged between these elements [18]. Service providers and toll chargers are required to implement the relevant interfaces in their OBUs and RSE respectively, as well as in their back-office systems. *Interface 1* is used for data exchange between the OBU and the service provider’s back-office systems. *Interface 2* is used for data exchange between back-office systems of the service providers and toll chargers. *Interface 3* is used for data exchange between toll chargers’ back-office systems and Roadside Equipment (RSE), including toll declaration and enforcement data. *Interface 4* is used for data exchange between a toll charger’s RSE and a service provider’s OBU.

EETS provision entails personal data processing, which must be carried out in compliance with the EU Directive 95/46/EC [19] and Directive 2002/58/EC [20].

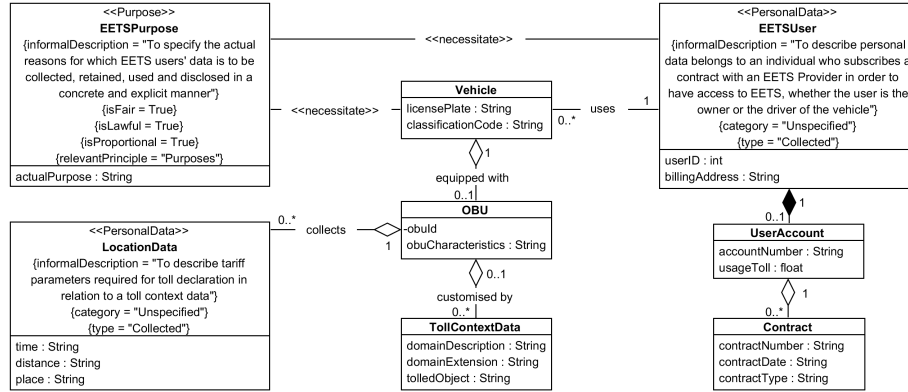


Fig. 6. EETSPurpose, along with the partial data model diagram for EETS.

5.2 An illustration

We now illustrate the APDL profile by means of our case study.

Purpose, PersonalData and DataModelling. EETSPurpose is a class stereotyped by «Purpose» to represent the main purpose for which EETS users' personal data is collected and processed. At the domain level, its tagged values informally describe the aim of the stereotyped class, as well as the fairness, lawfulness and proportionality of the purpose, and indicate the relevant GPS principle, as illustrated in Figure 6. At the instance level, the value of its actualPurpose attribute is to electronically calculate and collect road-usage tolls.

We draw a partial data model diagram that represents the following classes: EETSUser, LocationData, UserAccount, Contract, Vehicle, OBU and TollContextData, as illustrated in Figure 6. Classes that represent personal data are stereotyped by «PersonalData», whereas those represent generic data are not stereotyped. The relationships in which these classes participate can of course directly be modelled by associations in the UML.

As an example, EETSUser is a class stereotyped by «PersonalData» to represent the user's personal data, whether the user is the driver, owner, lesser or fleet operator of the vehicle. At the domain level, its tagged values informally describe the aim of the class, the category and the type of the data, as illustrated in Figure 6. Its primary attributes are userID and billingAddress.

DataLifecycle, LifecycleStage and its specialisations. In reference to the general architecture of the EETS proposed by the European Commission [18], the required personal data needs to be specified by Member States in relation to relevant national regulations. This indicates that the EETS data lifecycle is closed, i.e. no arbitrary data from external sources will be collected, acquired or derived without initial planning. In accordance with the physical nature of the EETS architecture, the collected EETS users' personal data is not stored in a centrally controlled infrastructure. Thus the EETS data lifecycle is decentralised.

EETSDataLifecycle is a class stereotyped by «DataLifecycle» to represent the main characteristics of the personal data lifecycle in the context of EETS. At the domain level, its tagged values informally describe the aim of the class, the openness of the processed data and the centrality of its underlying system, as illustrated in Figure 7. At the instance level, the values of its lifecycleType and granularityLevel attributes are Evolutionary and FinedGrained respectively.

Each lifecycle stage can be represented by one or more stereotyped classes that represent sets of activities according to: the types of processed data, its sources, the manner in which it is processed, and the assigned roles and responsibilities of the actors. This gives each lifecycle stage the ability to be expressed in terms of cycles, reflecting the repetitive nature of data processing activities. For example, SubscribingToService and CollectingUsageData are two classes of the collection stage, separated according to the nature of collection activities and responsibilities of the involved actors. To subscribe to an EETS service, drivers, as data subjects, provide their personal data by establishing and signing contracts with the service providers. To collect usage data, EETS service providers, as data processors, collect location data of subscribed EETS users via OBUs.

The SubscribingToService is a class stereotyped by «Collection» to represent a set of related activities, i.e. signing a contract and installing an OBU, with the aim of collecting users' personal data and vehicles' classification parameters. At the domain level, its tagged values informally describe the aim of the class, data sources, collection methods, available choices, consent type and the relevant GPS principle, as illustrated in Figure 7. At the instance level, the values of its createdUserData and createdVehicleData attributes are of the type EETSUser and Vehicle respectively.

The CollectingUsageData is a class stereotyped by «Collection» to represent a set of related activities, i.e. collecting location data, with the aim of collecting the time of usage, the covered distance and the place on which the vehicle is circulating on a particular toll domain for tolls declaration and calculation. At the domain level, its tagged values informally describe the aim of the class, data sources, collection methods, available choices, consent type and the relevant GPS principle, as illustrated in Figure 7. At the instance level, the value of its createdData attribute is of the type LocationData.

StageActivity, StageEvent and StageAction. Each lifecycle stage involves a set of stage activities, each of which contains a set of actions that represent its executable steps, and a set of events that cause the execution of these actions. SubscribingToService, for example, as a collection stage involves two stage activities: SigningContract and InstallingOBU.

SigningContract is a class stereotyped by «StageActivity» to represent the activity of collecting user's personal data by signing a contract with a service provider. At the domain level, its tagged values informally describe the aim of the class, the pre- and the post-conditions of the activity, as illustrated in Figure 8. At the instance level, the values of its input and output attributes are of the type EETSUser. The SigningContract coordinates its execution by containing

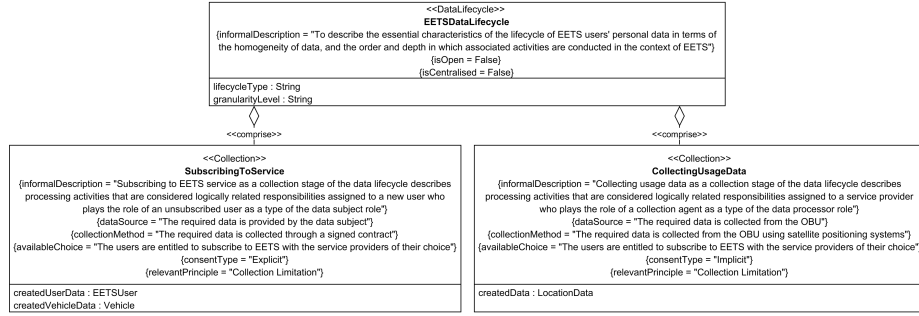


Fig. 7. EETSDataLifecycle and examples of its component stages: SubscribingToService, CollectingUsageData and CheckingOperationalCompliance.

CollectUserData as an action and defining Sign as an event that causes the execution of this action.

InstallingOBU is a class stereotyped by «StageActivity» to represent the activity of collecting vehicle’s classification parameters by initialising the OBU. At the domain level, its tagged values informally describe the aim of the class, the pre- and post-conditions of the activity, as illustrated in Figure 8. At the instance level, the values of its input and output attributes are of the type Vehicle. InstallingOBU coordinates its execution by containing CollectVehicleData as an action and defining Subscribe as an event that causes its execution.

LifecycleRole and LifecycleActor. Each lifecycle stage includes a set of lifecycle roles, each of which is played by different actors according to their capabilities and responsibilities. The SubscribingToService, for example, includes UnsubscribedUser as a lifecycle role and Driver as a lifecycle actor.

UnsubscribedUser is a class stereotyped by «LifecycleRole» to represent a type of the data subject role as a set of related activities, i.e. SigningContract and InstallingOBU, that are expected to be performed together for a certain task, i.e. subscribing to EETS service. At the domain level, its tagged values informally describe the aim of the class, and the main role type, as per Figure 8.

Driver is a class stereotyped by «LifecycleActor» to represent the individual capable of, and responsible for, performing the activities of SigningContract and InstallingOBU, i.e. grouped as UnsubscribedUser, with the aim of subscribing to the EETS service. The performance of these activities can be achieved by requesting the Sign and Subscribe events that cause the execution of the CollectUserData and CollectVehicleData actions respectively. At the domain level, its tagged values informally describe the aim of the class and the nature of the actor, as illustrated in Figure 8.

In summary, the approach has served as a preliminary acquisition step to capture all required concepts that support requirements analysis — a critical step in the system development lifecycle. It represents data-processing activities in a contextual and fined-grained manner that is amenable to risk analysis and compliance checking.

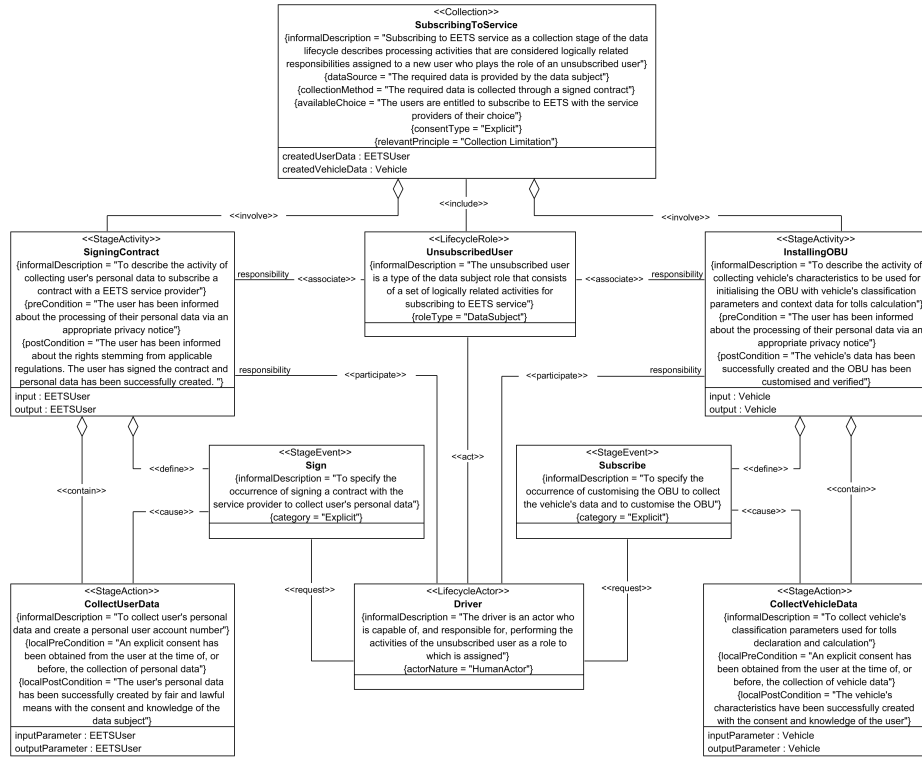


Fig. 8. SubscribingToService as a collection stage, along with its activities, actions, events, roles and actors.

6 Conclusions

The UML profile for the APDL model complements the contributions of [8] and [9], by providing foundations for analysing functional requirements and assessing potential privacy risks in a contextual and comprehensive manner. It represents privacy-related concepts using the standard extension mechanisms of the UML meta-model. Stereotypes and tagged values are used to represent key aspects of privacy principles as requirements, and constraints provide criteria for the evaluation of these aspects to determine whether the representation of data-processing activities fulfils these requirements.

The lifecycle stages and roles can be used to classify processing activities into multiple partitions, according to the nature of these activities, the capabilities and responsibilities of involved actors, the organisational units performing these activities or the geographical location at which these activities are performed. The APDL meta-model is a way of describing data-processing at a fined-grained level, with the possibility of expressing how activities are performed, what are their effects in terms of changes of states, when they take place in terms of

lifecycle stages, and where they take place in terms of lifecycle roles. This has the potential to support the application of the principle of separation of duties that manages the responsibility assignment by determining who is responsible for which lifecycle stage and what is the level of authority with respect to the decisions and activities performed when data is collaboratively collected and processed by multiple stakeholders in multiple domains. It also facilitates the application of the principle of data minimisation as a foundational step for privacy engineering by specifying the processed data items in each single atomic action within an activity. This, in turn, helps analyse and restrict the processing of personal data to the minimum amount necessary according to the purpose of each concrete action.

We will build upon this work in a number of ways. First, we will use additional case studies to further validate the approach and highlight its usefulness and practical impact in various domains. Possible examples include ePetitions [7]. Second, we will define a data-centric method to complement a PIA by analysing and assessing potential privacy risks in a comprehensive, contextual and non-reductive manner: comprehensive via the adoption of the APDL meta-model as a basis for contextual analysis; contextual via the adoption of Nissenbaum’s contextual integrity framework [14]; and non-reductive via the adoption of Solove’s taxonomy of privacy [17].

References

1. Alshammari, M., Simpson, A.C.: Personal Data Management for Privacy Engineering: An Abstract Personal Data Lifecycle Model. <https://www.cs.ox.ac.uk/publications/publication10942-abstract.html> (2017)
2. American Institute of Certified Public Accountants and Canadian Institute of Chartered Accountants (AICPA/CICA): Generally Accepted Privacy Principles. <https://www.cippguide.org/2010/07/01/generally-accepted-privacy-principles-gapp/> (2009)
3. Antignac, T., Scandariato, R., Schneider, G.: A Privacy-Aware Conceptual Model for Handling Personal Data. In: Margaria, T., Steffen, B. (eds.) *Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques*. Lecture Notes in Computer Science, vol. 9952, pp. 942–957. Springer (2016)
4. Balasch, J., Rial, A., Troncoso, C., Preneel, B., Verbauwhede, I., Geuens, C.: PrETP: Privacy-Preserving Electronic Toll Pricing. In: *Proceedings of the 19th USENIX Security Symposium*. pp. 63–78 (2010)
5. Cavoukian, A.: Creation of a Global Privacy Standard. <https://www.ipc.on.ca/images/Resources/gps.pdf> (2006)
6. Cavoukian, A.: *Privacy by Design ... Take the Challenge*. Office of the Information and Privacy Commissioner of Ontario (2009)
7. Diaz, C., Kosta, E., Dekeyser, H., Kohlweiss, M., Nigusse, G.: Privacy preserving electronic petitions. *Identity in the Information Society* 1(1), 203–219 (2008)
8. Gürses, S., Troncoso, C., Diaz, C.: Engineering Privacy by Design. *Computers, Privacy & Data Protection* 14 (2011)
9. Hoepman, J.H.: Privacy design strategies. *ICT Systems Security and Privacy Protection* pp. 446–459 (2014)

10. Joyee De, S., Le Métayer, D.: A Refinement Approach for the Reuse of Privacy Risk Analysis Results. In: Proceedings of the Fifth ENISA Annual Privacy Forum (APF 2017). pp. 73–109. Österreichische Computer Gesellschaft (2017)
11. Jürjens, J.: UMLsec: Extending UML for Secure Systems Development. In: International Conference on The Unified Modeling Language. pp. 412–425. Springer Berlin Heidelberg (2002)
12. Jürjens, J.: Secure Systems Development with UML. Berlin, Heidelberg : Springer-Verlag Berlin Heidelberg (2005)
13. Martín, Y.S., Del Alamo, J.M., Yelmo, J.C.: Engineering privacy requirements valuable lessons from another realm. In: 2014 IEEE 1st Workshop on Evolving Security and Privacy Requirements Engineering (ESPRE). pp. 19–24. IEEE (2014)
14. Nissenbaum, H.F.: Privacy in context: Technology, Policy, and the Integrity of Social Life. Stanford University Press (2009)
15. Object Management Group: OMG Unified Modeling Language (OMG UML). <http://www.omg.org/spec/UML/> (2015)
16. Oetzel, M.C., Spiekermann, S.: A systematic methodology for privacy impact assessments: a design science approach. European Journal of Information Systems 23(2), 126–150 (2014)
17. Solove, D.J.: A Taxonomy of Privacy. University of Pennsylvania Law Review 154(3), 477–564 (2006)
18. The European Commission: The European Electronic Toll Service (EETS): 2011 Guide for the Application of the Directive on the Interoperability of Electronic Road Toll Systems. http://ec.europa.eu/transport/themes/its/road/application_areas/electronic_pricing_and_payment_en (2011)
19. The European Union: Official Journal of the European Communities: Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. <http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:31995L0046&from=EN> (1995)
20. The European Union: Official Journal of the European Communities: Directive 2002/58/EC of the European Parliament and of the Council of 12 July 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector. <http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32002L0058&from=EN> (2002)
21. The European Union: Official Journal of the European Communities: Directive 2004/52/EC Of the European Parliament and of the Council of 29 April 2004 on the interoperability of electronic road toll systems in the Community. <http://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32004L0052R%2801%29&from=EN> (2004)
22. The European Union: Official Journal of the European Communities: Commission Decision 2009/750/EC of 6 October 2009 on the definition of the European Electronic Toll Service and its technical elements. <http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32009D0750> (2009)
23. The European Union: Official Journal of the European Union: General Data Protection Regulation. <http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32016R0679> (2016)
24. United States Department of Health, Education and Welfare: Secretary’s Advisory Committee on Automated Personal Data Systems: Records, Computers and the Rights of Citizens: Report. [Cambridge, Mass.]: [MIT Press] (1973)
25. Wright, D.: The State of the Art in Privacy Impact Assessment. Computer Law & Security Review 28(1), 54–61 (2012)

Table 1. The APDL Profile: stereotyped classes and associations.

Ref.	Stereotype	Base Class	Parent	Relates
1	«Purpose»	Class	—	—
2	«PersonalData»	Class	—	—
3	«Necessitate»	Association	—	1 \rightarrow 2
4	«DataModelling»	Class	—	—
5	«Require»	Dependency	—	2 \rightarrow 4
6	«DataLifecycle»	Class	—	—
7	«Undergo»	Association	—	2 \rightarrow 6
8	«LifecycleStage»	Class	—	—
9	«Comprise»	Association	—	6 \rightarrow 8
10	«Initiation»	Class	«LifecycleStage»	—
11	«Collection»	Class	«LifecycleStage»	—
12	«Retention»	Class	«LifecycleStage»	—
13	«Access»	Class	«LifecycleStage»	—
14	«Usage»	Class	«LifecycleStage»	—
15	«Disclosure»	Class	«LifecycleStage»	—
16	«Review»	Class	«LifecycleStage»	—
17	«Destruction»	Class	«LifecycleStage»	—
18	«StageActivity»	Class	—	—
19	«Involve»	Association	—	8 \rightarrow 18
20	«StageEvent»	Class	—	—
21	«Define»	Association	—	18 \rightarrow 20
22	«StageAction»	Class	—	—
23	«Contain»	Association	—	18 \rightarrow 22
24	«Fulfil»	Abstraction	—	18 \rightarrow 1
25	«Produce»	Association	—	22 \rightarrow 2
26	«Consume»	Association	—	22 \rightarrow 2
27	«Cause»	Association	—	20 \rightarrow 22
28	«LifecycleRole»	Class	—	—
29	«Include»	Association	—	8 \rightarrow 28
30	«Associate»	Association	—	18 \rightarrow 28
31	«Build»	Association	—	28 \rightarrow 4
32	«LifecycleActor»	Class	—	—
33	«Act»	Association	—	32 \rightarrow 28
34	«Participate»	Association	—	32 \rightarrow 18
35	«Request»	Association	—	32 \rightarrow 20