A Privacy-Aware Conceptual Model for Handling Personal Data*

Thibaud Antignac, Riccardo Scandariato, and Gerardo Schneider

Department of Computer Science and Engineering,
Chalmers | University of Gothenburg, Sweden.
thibaud.antignac@chalmers.se,
{riccardo.scandariato,gerardo}@cse.gu.se

Abstract. Handling personal data adequately is one of the biggest challenges of our era. Consequently, law and regulations are in the process of being released, like the European General Data Protection Regulation (GDPR), which attempt to deal with these challenging issue early on. The core question motivating this work is how software developers can validate their technical design vis-a-vis the prescriptions of the privacy legislation. In this paper, we outline the technical concepts related to privacy that need to be taken into consideration in a software design. Second, we extend a popular design notation in order to support the privacy concepts illustrated in the previous point. Third, we show how some of the prescriptions of the privacy legislation and standards may be related to a technical design that employs our enriched notation, which would facilitate reasoning about compliance.

Keywords: Privacy, Conceptual Model, Data Flow Diagrams

1 Introduction

Handling personal data adequately is one of the biggest challenges of our era. As smart objects equipped with sensors and network connectivity begin to populate our daily life and collect more and more data about our life style, opinions and preferences, we perceive an increased discomfort for the potential of privacy violations that are hanging over us. Therefore, law and regulations are in the process of being released, like the European General Data Protection Regulation (GDPR) [10], which attempt to deal with these shifting circumstances early on. At the same time, one of the commendable concepts that have emerged from the privacy research community is known as the *Privacy by Design* (PbD) principle [3], which is based on the idea that any personal data processing environment should be designed so that privacy is taken into account since the very beginning of the development process. In particular, PbD implies that privacy

* Published in *7th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation - ISoLA'16 (1); Track: Privacy and Security Issues in Information Systems*, volume 9952 of LNCS, pages 942-957. Springer, Oct 2016. DOI: http://10.1007/978-3-319-47166-2_65

concerns should be addressed as early as the requirement elicitation phase as well as when designing the software architecture.

However, there is a big disconnect between the technical concepts handled by software architects and the prescriptions stated by laws and regulations, including the upcoming ones. For instance, architects use boxes-and-arrows diagrams to conceptualise and describe how a software system is structured. Boxes represent high-level computational entities (like software component or sub-systems) while arrows symbolise the exchange of information between them. From a notation perspective, *Data Flow Diagrams* (or DFDs) represent a very popular option used to draw such architectural diagrams. DFDs are also a convenient representation should an architect choose to validate their software design, as popular threat analysis techniques (e.g., STRIDE [15]) make use of such diagrams as their starting point.

The law is generally written as normative texts stating citizens rights and obligations for legal entities with respect to information processing. Granted that the PbD principle is applied, the core question motivating this work is how an architect can validate their technical design vis-a-vis the prescriptions of the privacy legislation.

In this context, the present paper provides the following *contributions*. First, we outline the technical concepts related to privacy that need to be taken into consideration in a software design. In particular, we show that three complementary angles need to be addressed: data processing, data management and data accounting. Second, we extend the DFD notation in order to support the privacy concepts illustrated in the previous point. Such enriched models directly reduce the semantic gap between the design world and the privacy law. Third, and related to the previous statement, we show how many of the prescriptions of the legislation (like the GDPR) and standards (like ISO 29100) can be related to a technical design that employs our enriched notation. Therefore, we argue that the augmented models we propose in this paper serve as a stepping stone to (formally) reason about the compliance of a technical design with respect to the privacy legislation and standards. Clearly, the PbD approach requires a holistic re-thinking of the software development lifecycle. This has happened for software security to a large extent, which is a comparably more mature discipline than privacy[1]. Therefore, this paper has also the ambition to outline a roadmap for future research in order to actualise the tenets of the PbD principle.

The rest of this paper is organised as follows. Section 2 discusses how taking into account privacy impacts the software design. Section 3 proposes an extension to classical DFD making it possible to deal with the notions derives previously. Section 4 shows how this extension allows to address most of the commonly acknowledged privacy principles. Finally, Section 5 proposes new directions of research to improve further the state of privacy awareness and personal data protection in information systems before concluding by Section 6.

[1] See the "Building Security in Maturity Model" (https://www.bsimm.com/).

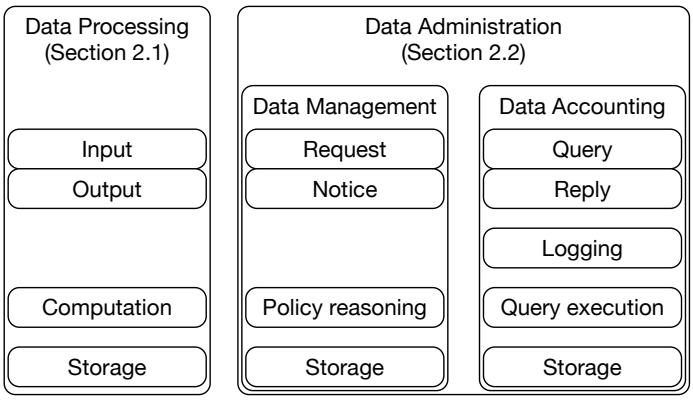

Fig. 1: Architectural view of an information system.

2 Privacy in Software Design

Beyond functional requirements, software designers are used to deal with non-functional ones for a long time, be them security, performance, or maintainability just to name a few. Compliance with personal data regulations and standards is another kind of non-functional requirement which strongly emerged in the recent past few years. As a result, designers have to take privacy into account from the beginning when they build information systems; a practice known as PbD. This calls for modelling techniques and tools equipped with privacy-related concepts in order to support these practitioners.

As shown in Figure 1, we propose to distinguish three different high-level facets of personal data processing, which are data processing (concerned with the actual data being manipulated), data management (concerned with the policy management of the system), and data accounting (aiming at logging and treating events occurring in the system). In what follows we describe how the handling of personal data can be viewed as lying on these three pillars.

2.1 Data Processing

Standards and regulations in general, and thus the personal data protection legislation in particular, do not rely on rigorous models. However, formally modelling the concepts at hand along with their properties is a necessary step to define what it means to *computationally* conform to a standard or to a regulation. This also helps to detect ambiguities that may come from informal reasoning[2]. We choose a model for such computations and show how it can be used in the ~~context of pe~~rsonal data handling in the following paragraphs.

[2] Law makers use techniques such as *legal drafting* which include a set of techniques and patterns to improve law consistency and clarity. However, they should be best regarded as best practices than as full-fledged analysis tools.

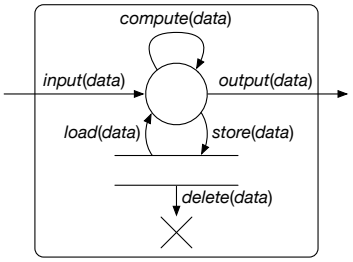

Fig. 2: Generic data processing.

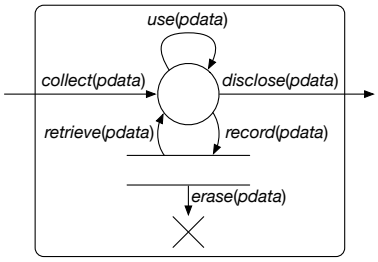

Fig. 3: Personal data processing.

Generic Data Many levels could be chosen to represent computation, ranging from very low-level (such as transistors in the processor) to very high-level (such as application level abstract state machines). There is no such a thing as a one-level-fits-all model and we have to choose our level of abstraction as the simplest one allowing to represent the properties we are interested in.

Figure 2 shows a high-level conceptual model for generic data processing. It models generic computations and is independent of personal data. In the figure, the circle represents a computational component while the element with two parallel lines represents a storage component. There are three kinds of activities a data processing system may perform on some *data*: communication, computation, and storage. The performance of these activities is modelled as the occurrence of events. Communications are modelled as *input* and *output* events. Computations are represented by *compute* events, and storage by *store* and *load* events. The *delete* event models data disposal.

We observe that computation cannot happen on stored data without it being loaded. Similarly, only stored data may be erased. This choice gives some freedom for the operation modes that can be modelled. For instance, it is possible to model on-the-fly data stream processing by not involving any *store* event.

Personal Data As outlined in the GDPR, personal data processing generally implies the following kinds of computations: collection, disclosure, usage, record, retrieval, and erasure. As shown in Figure 3, these privacy-sensitive events mirror the events in Figure 2, although they deal with personal data instead (*pdata*).

The definition of these notions with respect to a processing model allows us to make the link with normative prescriptions as they are stated in standards and regulatory texts. Indeed, personal data is subject to more obligations and duties for data controllers and data processors than generic data is (although what we consider as generic data here may actually be classified information, which may be subject to other kinds of sensitive data protection frameworks).

There is one more event related to personal data, not shown in the figure, which denotes (irreversible) anonymisation of personal data: *anonymise*. This event transforms some personal data *pdata* into generic (non-personal) *data*. When used, the data can be considered as free from any constraint deriving from personal data regulation.

The abstractions in Figures 2 and 3 enable us to specifically target personal data events in a model to check their conformance to specific provisions. For instance, it is generally considered, as a basic requirement, that personal data should be processed for a specific *purpose* to which the data subject has given his or her *consent*. Taking into account such constraints is the topic of the next section about data administration.

2.2 Data Administration

As shown in Figure 1, data administration consists of two main separated aspects which are data management and data accounting.

Data Management Operations on personal data need to be allowed, prevented, or triggered depending on their nature and on external requirements. In this paper we are particularly interested in the requirements about compliance to personal data regulations. The privacy provisions expected to be met are translated into *policies* allowing to manage data processing as defined above. These policies themselves are considered as generic or as personal data depending on their generality or whether or not they embed references to individuals. Thus they can also be matched to the data processing as defined in Figures 2 and 3. The *request* and *notice* components of the architecture in Figure 1 reflect the communication capabilities at the data management level, the *policy reasoning* component stands for the computation capabilities, and the *storage* component links to the storage capabilities. However, they have an additional power compared to the simple data considered in the previous section: they have effect on simple data with the purpose of enforcing policies. The enforcement of these policies is treated through reference enforcement mechanisms, similar to their security homologues.

Data Accounting Beyond data processing and data management, which constitutes the core of the activities of a data processing system, accountability is another recommended aspect to be taken into account. Data accounting is decomposed into two main activities consisting in: (i) logging some or all the events occurring during data processing, and (ii) allowing queries on the set logged events and providing replies accordingly. These communication capabilities are reflected by the *logging* and the *query* and *reply* components in Figure 1. Data accounting is also equipped with computation capabilities (offered by the *query execution* component) and storage capabilities (cf. the *storage* component).

3 Enabling Privacy in Software Design

In the previous section we introduced some of the basic concepts needed to model an information system while having privacy in mind, so that software designers are enabled to put PbD into practice. In this section, we show how to integrate these notions with a specific notation, namely DFDs [16]. We start by giving some preliminaries about DFDs before showing how they can be extended to become privacy-aware.

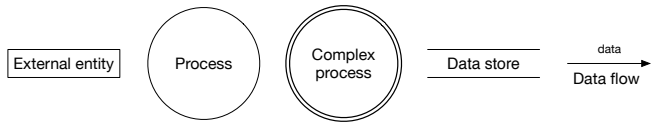

Fig. 4: Standard DFD notation.

3.1 Data Flow Diagrams (DFDs)

DFDs are a graphical notation which allows to model data flows in information systems. Their generality and modularity make them suitable for a large span of contexts and this is the main reason why they are widely used by practitioners to represent information systems architectures. As shown in Figure 4, the basic elements in a DFD are *external entities* (like users or third-party systems), *processes* (representing units of functionality), *data stores* (like data bases) and *data flows*. Complex processes can be refined into sub-processes and, in this case, are marked with a double edged circle. All data flows need to start from or to lead to a process (or to a complex process) for the data flow to be valid. For example, there cannot be a data flow directly from one data store to another.

Why Data Flow Diagrams? DFDs are already widely used in information security techniques. For example, the STRIDE threat modelling method developed at Microsoft [15] relies on a model of the software system expressed as a DFD. STRIDE is a risk-based security methodology aiming at eliciting controls and counter-measures when potential security flaws are discovered in a design. Another method, inspired by STRIDE and called LINDDUN [7], proposes a similar approach targeted at privacy threat modelling. The threats addressed by LINDDUN are linkability, identifiability, non-repudiation, detectability, disclosure of information, unawareness, and non-compliance. In this paper, we focus our attention on this last category of threats (i.e., non-compliance), which is only partially addressed by LINDDUN and largely delegated to the advise of a legal counselor. This work shows how the provisions dictated by standards or regulations can be handled on a technical level.

Example 1. Let us consider a Slippery Road Alert (SRA) system such as the ones proposed by car manufacturers[3]. A very simple DFD illustrating the main function of this system is presented in Figure 5. A car, modelled as an external entity, sends an SRA which is stored in an SRA database. These alerts are composed of the identifier of the car, a timestamp, a location, and a status. The alerts are then broadcast to all other cars. In practice, this broadcast should be limited to cars potentially impacted by the alert (i.e., in the same geographical area for instance). In this example, we keep it as simple as possible to focus on the privacy aspects as will be detailed later.

[3] https://www.media.volvocars.com/global/en-gb/media/videos/159534/slippery-road-alert-technology-by-volvo-cars5

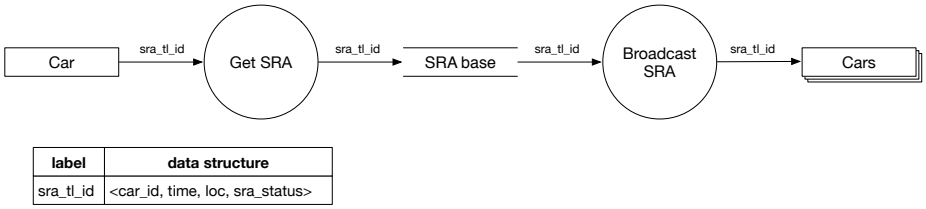

Fig. 5: DFD of a Slippery Road Alert system.

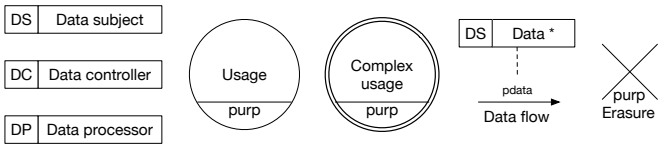

Fig. 6: Privacy-Aware DFD notation.

3.2 Privacy-Aware Data Flow Diagrams (PA-DFDs)

In order to enable designers to take privacy principles into account we extend standards DFDs with privacy-aware annotations. The new elements of a *Privacy-Aware DFD* (PA-DFD) are presented in Figure 6. We introduce three different kinds of external entities which are *data subjects*, *data controllers*, and *data processors* as these are the three main kinds of natural and legal persons discussed in regulatory texts. We also add the *usage* and *complex usage* process elements, which are annotated with a purpose *purp*. The element modelling a *data flow* of personal data is labeled with *pdata* and linked through a dotted line to the corresponding data subject the personal data refers to. We should note here that *pdata* embeds some useful metadata, as for instance the *purposes* to which the data subject has *consented*. More metadata may be added, such as the allowed *retention time* or the *age* of the data subject (specific provisions happen when the data subject is minor). Finally, we add a process element to model *erasure*. Such an element is also associated with a purpose *purp*. Adding a purpose to limit the erasure of data may seem useless, as deleting data could be considered a general good practice with respect to privacy. However, this may not be the case when data has to be stored without the consent of the data subject (for penal purposes for instance) or when the erasure could lead to a weakening of the rights of the data subject (a service provider could delete a contract to escape his obligations for example). Regulations expect all kinds of personal data processing to be associated with a purpose, which is reflected here in this choice.

The addition of these elements to DFDs allows to represent all the personal data processing events shown previously in Figure 3. Indeed, usage, storage, and erasure are first-class citizens of PA-DFDs, while the other events (collection, disclosure, recording and retrieval) can be represented as interactions of atomic events, as shown in Figure 7. A collection of data happens when some personal data (and its associated metadata) flows from an external entity. This personal

Fig. 7: Element interactions in Privacy-Aware DFDs.

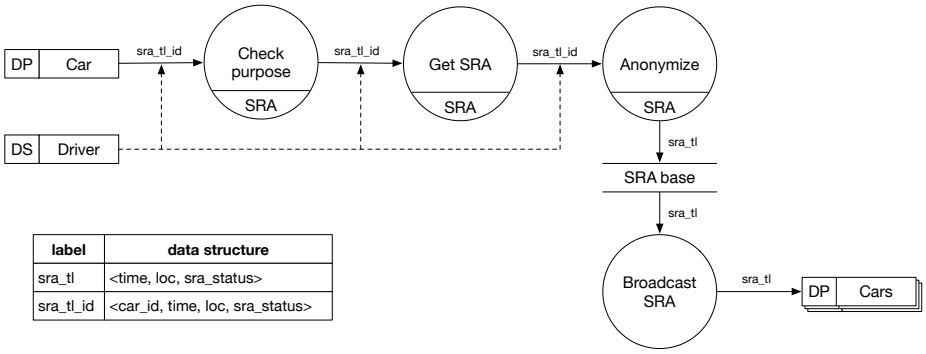

Fig. 8: PA-DFD of the Slippery Road Alert system from Figure 5.

data is linked to the corresponding data subject through the dotted line. Rules about valid PA-DFDs require the data flow to lead to a (complex) usage element (not shown here due to space consideration). The same applies for other kinds of personal data processing.

Example 2. We exemplify now how the extensions presented previously prove to be useful in the SRA system introduced in Example 1. Two processes are added as shown in Figure 8: (i) to check whether the purpose of use satisfies the consent given by the data subject (i.e., the driver), and (ii) to anonymise the data by pruning the car identifier. After anonymisation, the data is no longer considered as personal data, hence not linked to the data subject (and free from personal data regulations). More privacy aspects have to be taken into account for a full-fledged analysis. This example only illustrates how the notation is used.

Thanks to the privacy-aware extension, the designer's choices vis-à-vis privacy can be made explicit in the design model and analysed in a rigorous way. In next section we review several privacy principles and hint to how they fit the PA-DFD model.

4 Privacy Principles and Privacy-Aware DFDs

The definition of privacy is not unique, depends on cultures and contexts, and has varied in time since its first mention by Warren and Brandeis in 1890 [19]. Today, companies follow mainly two regulatory sources: (i) legal frameworks, which are defined by the law maker and thus vary among jurisdictions, and (ii)

standards, which are defined by the industry through consensus procedures and thus contain commonly agreed good practices. Though not imperative, standards have the two advantages of being closer to the views of industry and not to vary depending on jurisdictions. This is why we have chosen to review the privacy principles from the ISO 29100 standard, which is the most prominent standard in use today for this purpose. In the remainder of this section, we systematically consider and discuss each one of the ISO's eleven privacy principles.

Consent and Choice. The consent to personal data processing can be attached as metadata to personal data. The metadata should contain a set of purposes the data subject has consented to and against which the intent of a usage has to be checked. However, it should be noted that some data may be legitimately (and so lawfully) processed without the subject consent. In this case this is stated as a specific metadata allowing to bypass the consent/purpose satisfaction. Some aspects may be more challenging to model such as the fact that a consent has been given in a specific way (for it to be considered as freely chosen) for instance. This constraint, and others of the same kind, may be represented as boolean metadata, set to `true` by authorised parties such as a legal counsellor.

Purpose Legitimacy and Specification. Ensuring that the purpose complies with applicable laws or whether a piece of data is particularly sensitive is also out of the scope of the PA-DFD-based modelling. We will then also rely on boolean metadata attached to the personal data. On the other hand, by relying on the PA-DFD notation it is possible to model that the purpose of a processing element has been communicated to the data subject before collection by adding a notice in the data management part of the architecture.

Collection Limitation. The limitation of the collection can be divided into two aspects: one is legal and the other is technical. On the legal side, we model the lawfulness of the collection under relevant regulations as a boolean metadata which then has to be checked. The technical limitation ensures that only the data technically needed to fulfil the purpose is collected. This would require the analysis of the information system to check as follows: for each piece of personal data collected, there should be a usage element (or complex usage) in the PA-DFD diagram that takes it as an input. Note that this can be done syntactically or semantically, depending on the precision desired and the tools available[4].

Data Minimisation. Data minimisation is divided into four different concerns:

- The first is about minimising the quantity of data processed and the number of people having access to it. In the PA-DFD model, disclosures and retrievals are clearly identified and each of them should be questioned about its necessity by an expert. This may be instantiated through a set of questions for which the replies given by the designer (or an expert) are traced.
- The need-to-know principle also has to be checked by an expert. This is because it is linked with the semantics of the data itself, and the PA-DFD

[4] Many dependency analysis rely on syntactic dependence, which is weak from a security standpoint, but reasonable for debugging cases to support the designer when it is assumed there is no voluntary attack.

model only addresses the information system architecture. Such principles could also be addressed by having a validated semantic ontology linking data to purposes which the analysis could rely on.
- Privacy by default has to be ensured at the application level, and thus it is defined as a boolean metadata attached to the data (also validated by an external expert or by the use of an approved library or collection of patterns).
- Erasure of personal data, as soon as the retention period expires, is a requirement that can be modelled relying on the time dimension of the model.

Use, Retention, and Disclosure Limitation. Limitations are also divided into four different concerns according to the ISO standard:
- Data minimisation applied to the whole data lifecycle is similar to Collection Limitation, as described above. Thus the same way to proceed is chosen.
- Limiting the use of personal data to purposes specified by the controller is done as for the Consent and Choice principle described above.
- Retaining data as long as needed is a global property. We should check whether the data could (legally) be processed in the future. If we model future potential usages it would be possible to determine whether the data should be retained. However, this is difficult to determine. Also, the choice between deleting and anonymising data could be based on whether there is a need to anonymise part of the data. For example, if aggregated statistics are expected then date should be retained and anonymised.
- Legal obligations to archive personal data can be taken into account by preventing an erasure of these data (and so should be globally verified for the modelled system). Moreover, the data accounting part of the architecture may play a role here through the logging of personal data processing.

Accuracy and Quality. Accuracy and quality is mainly treated as a boolean metadata as it calls other subtasks, such as verifying the semantic quality of the data before processing it. However, an implicit implication is the consistency of the data in the model. If some personal data is stored at different locations, the data itself should be consistent. This can be tracked by verifying that modifications on data retrieved from a data store are propagated to other data stores where this data may also reside (as a global property of the information system).

Openness, Transparency, and Notice. These principles are subdivided into four different facets as follows:
- A mechanism to release policies to data subjects is implemented at the data management layer through the request/notice mechanism. This mechanism allows communication about data processing between the data controller and the data subject. We assume all policies are public (maximum transparency) though it would not be difficult to add some guards to restrict the communication of the policy and thus restrict the transparency in case of a trade's secret (as it is an allowed exception in some regulations for instance).
- Notifications to the data subject when her data is being processed, detailed data such as the purpose, the stakeholders involved, and the identity of the controller, are also addressed by the data management part of the architecture which should systematically send notifications before data is processed.

– The data controller also has to inform the data subject about the actions she may have taken concerning her data, and about the means to actually exercise her control. This is somehow similar to an interface notification. We assume this is standardised, and addressed by the data management layer.
– Notification of major changes in procedures, however, is hard to model as it would imply being able to dynamically detect evolutions in the data processing layer of the information system. This is clearly out of reach of the notation we are defining. Here, PA-DFDs only model static architectures.

Individual Participation and Access. Four points to be taken into account:
– Access and reviews of personal data by the data subjects are handled by the request and notice parts of the data management component. Authentication is assumed to be performed by a suitable additional component, and the absence of any prohibition to access such data by the data controller should be attested by a boolean metadata.
– The right to modify the data handled by the data controller is also enforced by the data manager. Through requests, the data subject may express requirements for changes on her data. These have to be forwarded and enforced adequately by triggering the relevant events in the data processing layer.
– The preceding item should also be notified to third-parties through their request/notice interface to ensure the required changes are applied.
– Procedures should be created to enable data subjects to exercise their rights in an easy way.

Accountability. Modelling the information system with a PA-DFD is a major advance in term of accountability as it shows demonstrably that many decisions have been taken along with their motivations (provided it has then been implemented correctly). One important point is the presence of ways to complain for the data subjects and to redress issues if needed. This can be modelled through dedicated usage elements in the data management part of the architecture and triggered through the request/notice interface. Similarly, privacy breaches in the system should be notified to the data subject by an appropriate mean.

Information Security. Our modelling assumes the events are performed as they should (meaning securely). Security issues should be part of a security analysis (such as STRIDE) and are not modelled here. However, security breaches may be notified by the notification mechanism as mentioned earlier. Moreover, if we want to model security attacks, we could add an event *corrupt* for integrity, and *reveal* for confidentiality attacks, to the data processing in Figure 3.

Privacy Compliance. Modelling an information system with PA-DFD will help to improve privacy compliance. Note that the ISO standard contains many more aspects also covered by regulations and that are not easy to be directly addressed by technical means. The most conservative approach is to add boolean metadata and associated guards for each event that should be reviewed before being performed. Some of these verifications could be automated and integrated into the model at a later stage. An important part of privacy compliance focuses on *accountability* through the data accounting layer, as described earlier.

Example 3. We now discuss how the privacy-aware entities introduced in Example 2 improve the privacy compliance along the privacy principles just described. The purposes added to the processes consists in the purpose specification of the personal data processing. They are used to guarantee consent respect. Use of anonymisation allows to minimise the amount of personal data stored and processed. Though not complete w.r.t. the ISO standard, these aspects improve privacy compliance and illustrate how PA-DFD help designers to adopt PbD.

5 Roadmap for Research

Modelling a system while relying on the approach presented here calls for research in two orthogonal dimensions: i) Methodologies to build a PbD architecture; ii) Tools and techniques used during the data lifecycle, including both foundational as well as practical aspects, in order to mitigate attacks and non-compliance. We describe below a non-exhaustive list of research directions along these two different axes.

Privacy by Design

PbD is still a promise to be fulfilled and challenges abound (see for instance [18] and the recent ENISA[5] reports [5, 6]). To address the privacy principles detailed in Section 4, we have argued for the need of a more rigorous model allowing to describe the different facets of personal data handling as shown in Figure 1. The need for formal models for privacy has already been discussed in the literature (e.g., [1, 17]). We have here proposed to rely on enriched DFDs (cf. Figures 6 and 7), due to their acceptance in industry. One clear research direction is to enhance these privacy-aware DFDs with formal semantics. Different approaches may be envisaged for this purpose depending on the kind of analysis one might be interested on. One possibility would be to use *Petri nets* [14], in particular the *Timed-Colored* variant [11]. Indeed, data flowing in the information system can be encoded as (colored) tokens, while conditions (including timing constraints) to be fulfilled may be modelled by guards on transitions. Such tokens can hold any arbitrary data, including time, allowing for extended policy reasoning.

The advantages of giving a formal semantic will not only make more precise to software engineers about the meaning of what they write, but also will give the possibility to reduce the number of design errors taking advantage of formal verification. Some properties might be enforced by construction while editing. For instance, while designing the architecture, the tool could warn the engineer on a missing check for a data containing a retention time constraint, or when deleting data on that these should be done on every component if the data is being forwarded somewhere else. Other properties could be verified *a posteriori*. For instance, the enforcement of data retention policies implies that there should not be any reachable state in which the data is still stored after expiration of the retention time (a *safety* property). A *liveness* property would be checking that a data subject should always be able to access or rectify the data about herself.

[5] *European Union Agency for Network and Information Security.*

The model presented in this paper allows to model personal data processing and how it interacts with privacy principles prescribed by industrial good practices. However, these prescriptions give very few indications to the designer about how to actually build a model such that they are met. So, another research direction concerns the need of a methodology to guide the designer through the multiple steps leading to a satisfactory architecture. The model presented here makes it easier for the designer as each data processing and data management unit is self-sufficient with regard to the satisfaction of some privacy principles. However, some other principles are global and cannot be achieved by a simple composition of conformant sub-components, such as data minimisation for instance. To this aim, a computer-aided design tool should help and guide the software designer not to include specific privacy issues in relevant places of the design. This diagram-based formalism could be enriched with automatic pop-up windows while editing the architecture to remind the designer that the data minimisation should be checked.

Attacks and Non-Compliance

We look here at where improper personal data processing may occur during the data life cycle. Before describing some concrete research directions let us clarify why we split the possible issues into two kinds, *attacks* and *non-compliance*. This distinction relies on the activity or passivity of the parties, corresponding to two complementary dimensions: security and privacy, respectively.

Non-compliance captures the cases when a flaw happen without any active behaviour targeting the flaw. On the other hand, *attacks* are intentional actions. With regard to attacker models frequently encountered in the computer security literature, even passive attackers (also called honest-but-curious), are considered to belong to the attacks category because the unexpected computations themselves are positive actions taken by the attacker. Another difference is that in the case of attacks, there is always an attacker (or a group of attackers) while this is not the case for non-compliance. In general, a security attack involving personal data implies some form of personal data non-compliance. Moreover, information security is considered as one of the privacy principles to meet. Some security attacks do not involve personal data and are addressed elsewhere in the literature and will thus not be described here.

Issues from privacy non-compliance come from technical policies and implementation which do not conform to provisions (stated in regulations, laws, terms, contracts or binding corporate rules). Such non-compliances may occur at any step of the lifecycle. They can be defined by any violation of the regulatory prescription. Some of them apply to any computation on personal data (known as processing) while others apply only to some parts of the data life cycle.

When collecting data an important issue is that of *data minimisation*, i.e. that a program does not use more data than needed for the intended purpose. To the best of our knowledge not much research has been done in what concerns how to guarantee that a program satisfies this principle. To start with, there is a need to do some foundational research concerning minimisation: What does it mean, formally? The idea would be to determine what are exactly the needed

input to compute each possible output. This notion seems to be related to other concepts in information-flow security like *non-interference* [4]. It would be interesting to spell out the connection, if any. How to build programs satisfying data minimisation by construction? Or at least to find out how to check whether a program satisfies the principle. To find suitable answers to these questions is an interesting research direction.

Accountability is another important issue mostly now when third-party software is everywhere and finding liabilities is not easy. In the presence of a privacy breach, who is to be held responsible? How to find the component causing the breach, and thus the ultimate responsible? Keeping a log of all the system events seem an attractive solution, and might somehow help here. That said, keeping a log file could by itself be the cause of a privacy breach. A challenge here is to find good ways to log all the events of a system while protecting the log file, developing strong authentication techniques to control who may access it.

Cross-cutting the different phases in the personal data life cycle is that of ensuring that private data is well managed across multiple parties. Though our approach does try to help designing software that takes this into account, there still is the possibility that data is misused. One way to complement the PbD approach is to consider different kinds of runtime checking. In particular, one concrete interesting approach is *sticky policies*: attaching policies to data in order to constrain their use so the policies are respected. Some work has been done in this direction, notably in the context of the EnCoRe project [13], but more research is needed in order to make the approach part of the PbD process.

A different aspect of privacy is that of making queries to multiple databases and making inferences by aggregating information. One recent promising technique to guarantee that (some quantity of) privacy is preserved is *differential privacy* [8]. The theory establishes that by adding the right amount of noise to statistical queries, it is possible to get useful results at the same time as providing a quantifiable notion of privacy. Adding noise raises the issue of usability vs. privacy. Though there are few prototypical tools based on differential privacy, e.g. PINQ implementing the original idea [12], and the tool ProPer refining PINQ to individual records [9], there still is a gap between theory and practice [2]. More research is needed to make differential privacy practical.

6 Conclusion

In this paper we have outlined PA-DFD, a conceptual model based on an extension of DFDs with privacy concepts taken from the GDPR and the ISO 29100 standard. As discussed in Section 4, not all the privacy principles of the ISO standard are captured by our model, but it is encouraging to see that we do capture most of them in a direct way, while other principles can be captured by using added metadata. We are aware that our proposal does not solve all privacy issues related to design. However, our aim is to provide a first step towards a systematic precise methodology (supported by tools) to help software engineers to design software having privacy in mind. We envisage a tool where the engineer

writes a DFD focusing only on the functional aspects, and the tool would automatically generate the PA-DFD. As discussed in Section 5, a lot remains to be done. We hope that more researchers and practitioners continue to address those (and other) issues to help towards suitable solutions to the privacy problem.

Acknowledgements This research has been supported by the Swedish funding agency SSF under the grant *DataBIn: Data Driven Secure Business Intelligence*.

References

1. Abe, A., Simpson, A.: Formal models for privacy. In: EDBT/ICDT Workshops. CEUR Workshop Proc., vol. 1558. CEUR-WS.org (2016)
2. Bambauer, J., Muralidhar, K., Sarathy, R.: Fool's gold: An illustrated critique of differential privacy. Vanderbilt J. of Entert. & Tech. Law 16(4), 701–755 (2014)
3. Cavoukian, A.: Privacy by design: Origins, meaning, and prospects. Privacy Protection Measures and Technologies in Bus. Org.: Aspects and Standards 170 (2011)
4. Cohen, E.: Information transmission in computational systems. SIGOPS Oper. Syst. Rev. 11(5), 133–139 (1977)
5. D'Acquisto, G., Domingo-Ferrer, J., Kikiras, P., Torra, V., de Montjoye, Y.A., Bourka, A.: Privacy by design in big data. ENISA Report (December 2015)
6. Danezis, G., Domingo-Ferrer, J., Hansen, M., Hoepman, J.H., Le Métayer, D., Tirtea, R., Schiffner, S.: Privacy and data protection by design. ENISA Report (January 2015)
7. Deng, M., Wuyts, K., Scandariato, R., Preneel, B., Joosen, W.: A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements. Requirements Engineering 16(1), 3–32 (2010)
8. Dwork, C.: Differential privacy. In: ICALP'06. LNCS, vol. 4052, pp. 1–12. Springer Verlag (2006)
9. Ebadi, H., Sands, D., Schneider, G.: Differential Privacy: Now it's Getting Personal. In: POPL'15. pp. 69–81. ACM (2015)
10. European Commission: Proposal for a General Data Protection Regulation. Codecision legislative procedure for a regulation 2012/0011 (COD), European Commission, Brussels, Belgium (January 2012)
11. Jensen, K., Kristensen, L.M.: Coloured Petri nets: modelling and validation of concurrent systems. Springer Science & Business Media (2009)
12. McSherry, F.D.: Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In: ACM SIGMOD'09. pp. 19–30. ACM (2009)
13. Pearson, S., Mont, M.C.: Sticky policies: An approach for managing privacy across multiple parties. IEEE Computer 44(9), 60–68 (2011)
14. Petri, C.A.: Kommunikation mit Automaten. Ph.D. thesis, Institut für instrumentelle Mathematik, Bonn (1962)
15. Shostack, A.: Threat modeling: Designing for security. John Wiley & Sons (2014)
16. Stevens, W.P., Myers, G.J., Constantine, L.L.: Structured design. IBM Systems Journal 13(2), 115–139 (1974)
17. Tschantz, M.C., Wing, J.M.: Formal methods for privacy. In: FM'09. LNCS, vol. 5850, pp. 1–15. Springer (2009)
18. Tsormpatzoudi, P., Berendt, B., Coudert, F.: Privacy by design: From research and policy to practice - the challenge of multi-disciplinarity. In: APF'15. LNCS, vol. 9484, pp. 199–212. Springer (2016)

19. Warren, S.D., Brandeis, L.D.: The right to privacy. Harvard law review pp. 193–220 (1890)