A Metamodel for Privacy Engineering Methods

Yod-Samuel Martín
Departamento de Ingeniería de Sistemas Telemáticos
Universidad Politécnica de Madrid
Madrid, Spain
samuelm@dit.upm.es
ORCID: 0000-0002-0065-5117

José M. del Álamo
Departamento de Ingeniería de Sistemas Telemáticos
Universidad Politécnica de Madrid
Madrid, Spain
jm.delalamo@upm.es
ORCID: 0000-0002-6513-0303

***Abstract*— Engineering privacy in information systems requires systematic methods to capture and address privacy issues throughout the development process. However, the diversity of both privacy and engineering approaches, together with the specific context and scope of each project, have spawned a plethora of privacy engineering methods. Method engineering can help to cope with this landscape, as it allows describing existing methods in terms of a limited variety of method elements (and eventually enable their recombination into new, customized methods). This paper applies method engineering to introduce a privacy engineering metamodel, whose applicability is illustrated with a set of popular privacy engineering method elements, and a widely recognized privacy engineering method.**

Keywords*— *Privacy engineering metamodel; Method engineering; Privacy engineering; Privacy Methods; Methodology; Metamodel; ISO/IEC 24744; SEMDM; Privacy by Design; GDPR; LINDDUN

I. INTRODUCTION

Despite the increasing urgency in addressing privacy concerns associated with information systems, and the technical developments available, engineering privacy-friendly systems remains a challenge for several reasons. First, privacy is a multi-disciplinary, essentially contested concept [1], which can thus be subject to multiple reference frameworks, be them social, legal, or technical. Second, research efforts have focused on tackling privacy issues by technical means, rather than investing in generalizing and systematizing the application of said technical solutions so that others can reuse and apply them. Third, even when a given privacy framework is set, the diversity of information systems (platforms, APIs, services, infrastructures, enterprise systems…) and development process models (agile, waterfall...) makes it difficult to elaborate a one-size-fits-all privacy engineering method[1].

In this context, dozens of novel contributions in the field of privacy engineering appear every year (of which the papers presented at this workshop represent a sample), each of which targets specific aspects and suits different situations. In order to assess the benefit and adequacy of any such solution, it would be desirable to have the relevant knowledge systematically organized so as to ease the communication within the community of practice and research of privacy engineering.

[1] For our purposes, we use both terms 'method' and 'methodology' interchangeably.

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731711.

This paper describes our contribution to this effort, by presenting a conceptual framework which allows arranging the different concepts that usually underlie the various contributions subsumed under the field of Privacy Engineering. This framework has been realized as a metamodel which extends SEMDM (the metamodel for software and systems development methodologies described in ISO/IEC 24744 [ISO24744]), and provides a controlled vocabulary of privacy engineering methodological elements and a normalized set of connection points and relationships to organize those elements. Thus, it enables the description of different elements of existent privacy engineering methods in comparable terms, so that they can be further catalogued and assessed. That metamodel can be thought of as a labelled rack, to each of whose compartments the contributions on privacy engineering can be anchored. Moreover, by enriching the description of method elements with well-defined connection hooks, their reuse and integration is fostered.

The remaining of the paper is structured as follows. Section II provides some background on privacy and method engineering. Then, section III describes our proposal for a privacy engineering metamodel based on the extension of the SEMDM metamodel, and section IV validates its applicability by constructing a representation of LINDDUN, a well-known privacy engineering method. Finally, section V discusses the potential applicability of the metamodel we propose so as to promote the reuse of privacy methodologies, and section VI concludes by summarizing the significance of our solution and pointing towards future work to overcome its limitations.

II. BACKGROUND

A. Privacy engineering

Privacy engineering is a nascent field of research and practice which pursues systematic approaches for the inception and application of privacy-oriented solutions throughout systems and software development processes. According to one of the first definitions of privacy engineering [2], the keystones of the field are:

- *Theories*, which deal with privacy from different approaches. For instance, for different authors, privacy may be a matter of non-intrusion, seclusion, limitation, control, boundary regulation, system architecture, policy or interaction, just to mention a few. Following that line, we consider that all privacy theories are born valid to apply privacy engineering, but different theories provide different conceptual frameworks to which the elements of each privacy-engineering method adhere.

- *Methods*, that is, processes for capturing and addressing privacy concerns during any of the stages of the lifecycle of information-based systems, which include their conception, development, management, and maintenance. Methods provide directions and rules and help set privacy goals, structured in a systematic way into tasks and stages with the aid of supporting tools and techniques.
- *Techniques*, which refer to procedures, possibly with a prescribed language or notation, to accomplish specific privacy engineering tasks.[2]
- *Tools*, that is, means (automated or not) that support privacy engineers in carrying out their responsibilities within a privacy engineering method.

Efforts on privacy engineering usually stick to the "Privacy by Design" (PbD) paradigm [3] which summons engineers and other stakeholders to integrate privacy aspects into the different activities they are involved, throughout the whole development lifecycle of information-based systems, rather than introducing them as an afterthought. Several PbD methods have been developed which define engineering activities that introduce privacy at different stages of the development lifecycle, defining what are usually named whole-lifecycle privacy methodologies.

One such effort exemplifies all the concepts described above: LINDDUN [4] is a privacy-engineering method focused on the privacy assessment of information systems. It conceptualizes privacy as seven distinct properties widely recognized by the privacy research community, represented by its corresponding threats (from whose initials LINDDUN takes its name). This method describes a set of techniques to e.g. identify privacy threats, and provides a tool called "threat catalogue" that supports privacy engineers on this process (v. section 4 below for a detailed description of LINDDUN in terms of our privacy engineering metamodel).

Although this conception of privacy engineering seems clearly founded, there is no common standard framework which privacy engineering developments may refer to. As a matter of fact, efforts on standards for privacy have long been undertaken by ISO/IEC JTC1/SC27/WG5 (Joint Technical Committee 1 of the ISO and the IE, subcommittee 27 on IT Security Techniques, working group 5 WG5 on identity management and privacy technologies), which has delivered general references that engineers and managers addressing privacy must take into account with regards to terminology, institutionalization of practice (i.e. ensuring that organizations apply the same good practices), and support for evaluation (i.e. approaches on how privacy is evaluated). However, those efforts only provide partial views of privacy engineering, as they deal with individual perspectives (e.g. privacy principles, best practices, organizational maturity), isolated processes (e.g. impact assessment, requirements analysis), or specific domains (Big Data, Internet of Things). Yet they lack a shared, all-encompassing conceptual framework, independent from specific privacy engineering methodologies, development practices and application domains.

It is under these circumstances that we introduce our approach, which proposes the definition of such a framework where the different privacy engineering methodologies may be pegged out, by formalizing the definition of a metamodel which consists of the elements that usually appear in privacy engineering methodologies and the relationships between one another.

B. Method engineering and SEMDM

Our approach is grounded in the discipline of method engineering, which is [5] "*the engineering discipline to design, construct and adapt methods, techniques and tools for the development of information systems*"; and which focuses on designing methods (or methodologies) for specific situations (e.g. a specific organization, a specific project) rather than resorting to rigid, existent methodologies.

Any methodology that may be constructed responds to an underpinning metamodel, i.e. an abstract model that describes the concepts that may be present in the methodology (i.e. the types of elements it is made of) and their potential relations with one another. Many methodologies may exist that conform to the same, shared metamodel (i.e. their descriptions can rely on a common set of terms).

One such metamodel is defined by the Software Engineering Metamodel for Development Methodologies (SEMDM)[7], standardized as ISO/IEC 24744 [8], and "*aimed to the definition of methodologies in information-based domains, i.e. areas characterized by their intensive reliance on information management and processing, such as software, business or systems engineering.*"

SEMDM proposes three layers of abstraction to define and instantiate methodologies: metamodel, methodology and endeavor (a.k.a. project). The metamodel defines the elements that methods engineers employ to enact methodologies. In turn, developers use the methodologies to construct products or deliver services in the context of particular endeavors.

The SEMDM metamodel describes a set of concepts that can be part of any methodology, and which cover the three dimensions of processes, producers (including people) and products:

- *Work Units* describe things to be executed, such as a *Process* (large-grained *Work Unit* that operates within a given area of expertise), a *Task* (a small-grained work unit that focuses on what must be done), or a *Technique* (a small-grained work unit that focuses on how it has to be done). A methodology may recommend using specific techniques for a task, with different degrees of *Recommendation* (e.g. compulsory, optional, discouraged, etc.).
- *Producers* describe agents that have the responsibility to carry out work units, and can be specialized into

[2] Note how these Techniques are methodological rather than technological, and hence they are different from the "Privacy Enhancing Technologies", which belong to the realm of the technology applied at each endeavor.

Roles (a collection of responsibilities that a producer can take), *Tools* (an instrument that helps another producer to execute its responsibilities in an automated way) and *Teams* (a set of producers).

- *Work Products* are artefacts of interest for the project which can be used as inputs, intermediate results, or outputs of a work unit; e.g. *Document*, *Software Item* or *Model*. In SEMDM, a *Model* provides an abstract representation of some modelling elements by aggregating a set of *Model Units*, which may be related with one another according to the grammar defined in a *Language*. Work products may be acted upon by work units through different *Actions* (read, create, modify, or delete).

- *Stages* represent a managed time frame with a specific objective within a project, either instantaneous such as a *Milestone*, or with duration such as a *Phase* (during which the same cognitive framework prevails), a *Build* (aimed delivering a version of a work product), or a *Time Cycle* (aimed at delivering the final product or service).

A specific methodology (or a method fragment) will define its own set of *Work Units*, *Producers*, *Work Products*, or *Stages* (e.g. a method may define a type of *Document* called Requirements Specification to be produced when the method is enacted). Each of these elements defined in a methodology holds a dual reality: on the one hand, they are instances of the concepts defined in the metamodel (e.g. Requirements Specification is a kind of *Document*); on the other one, they are *Templates* that will be instantiated at each endeavor when the methodology is enacted (i.e. each project will fulfill its own instance of Requirements Specification, according to the said *Template*). The relation between both perspectives is called a 'powertype' relationship. (The interested reader can get further details on the use of the powertype pattern for methodology metamodeling from [9].)

SEMDM also defines *Resources*, that is, methodology elements that are used 'as is' at the project level, without requiring any instantiation, namely:

- *Languages* which define a set of *Model Unit Kinds* focused on one modelling perspective, and the relations allowed among *Model Units* of those kinds. A *Language* can be any complex, structured system of related symbols able to convey meaning (which encompasses the formal languages that underlie programming languages, visual languages and natural languages alike, but also conceptual languages as in 'the language of art').

- *Notations* which are associated to a *Language*, and provide a concrete syntax to represent *Models* conforming to that language.

- *Guidelines* which tell how to use some methodology elements.

- *Constraints*, that is, conditions that hold or must hold at certain point in time within the methodology.

Thus, SEMDM provides a comprehensive and extendable metamodel for system and software method engineering. We have leveraged these features to develop a comprehensive privacy engineering metamodel (detailed in the next section) able to cope with the variety in privacy frameworks, business domains, types of systems, and development processes.

III. Privacy Engineering Metamodel

If privacy engineering promoters want it to be actually adopted, they cannot aim at proposing completely new methodologies from scratch which are disconnected from current practice. Rather, privacy engineering needs to be aligned with more general efforts on software and systems engineering, in order to ensure a smooth integration and ease its application and utility.

An important step in this direction would be the formulation of a conceptual framework that focuses on generalizing and systematizing privacy engineering methodology elements, so that they can be compared, assessed, and integrated.

Method engineering (and SEMDM in particular) provides an especially suitable foundation to model the field of privacy engineering. Indeed, the aforementioned definition of privacy engineering matches quite well the SEMDM metamodel, as both consider the notions of theories, methods, techniques and tools; to which SEMDM adds other concepts such as tasks, stages, roles, teams and work units, which are also relevant to privacy engineering methods, as we will show later. Besides, SEMDM addresses both the process and the product dimensions, both which are tackled by privacy engineering methodologies.

In consequence, we have built on and extended SEMDM to propose a *Privacy Engineering Metamodel*, by identifying a set of extensions to the standard SEMDM metamodel that support the concepts specific to privacy engineering. The standard SEMDM metamodel together with our extensions can be used to describe any privacy-engineering methodology. Of course, this can be further extended by specific privacy frameworks which refine these concepts or define their own extensions.

The most relevant extension to SEMDM that allows dealing with privacy engineering aspects consists in the definition of several types of *Resources* (in grey, Fig. 1) present in many privacy engineering methods, and which provide the foundations to deal with privacy engineering from different perspectives, namely ontological (*Privacy Conceptual Model*), deontological (*Privacy Normative Framework*), situational (*Privacy Engineering Code*), and epistemological (*Privacy Knowledge Base*). Besides, an abstract type of role (*Privacy Engineering Role*) subsumes the common responsibilities that may be expected from privacy engineers.

A. Privacy Conceptual Model (PCM) and Units (PCUs)

A *Privacy Conceptual Model (PCM)* provides a *conceptual description* of what 'privacy' is in the context of the privacy theory where a specific privacy-engineering methodology is grounded. Due to the plurality [10], contextuality [11], and contestability [1] of privacy as a social, political and legal concept and its different translations to the technical domain,

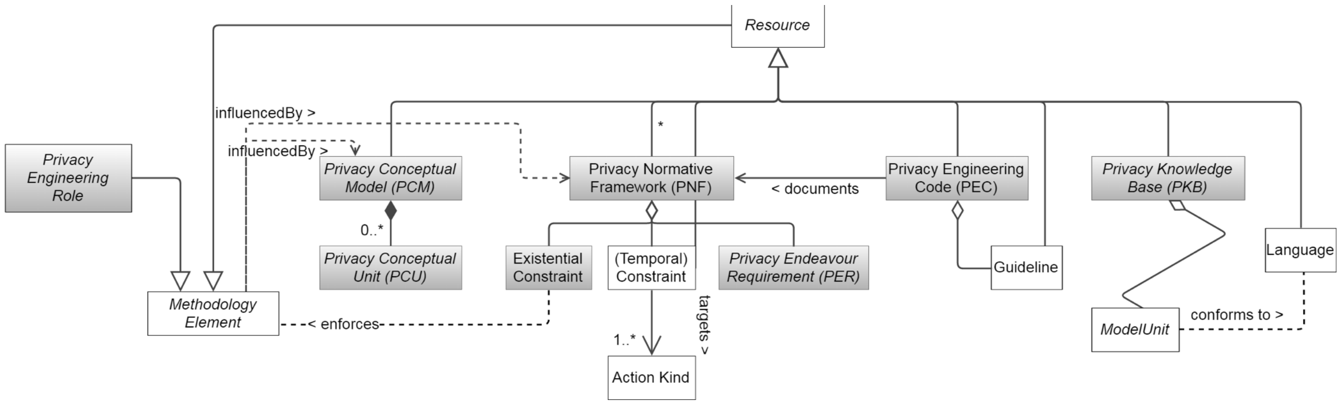

Fig. 1. Structural meta-model of the SEMDM extensions for the Privacy Engineering Framework

we refrain from folding a specific conception of privacy into the definition of our Privacy Engineering Metamodel. We do however assume that any defendable privacy engineering method will draw on some ontological definition of privacy, which may in turn influence all the methodology elements.

Besides providing a definition for privacy, the PCM may answer other questions such as what its subject and object are (cf. [1]). For instance, ISO 29100 privacy framework [12] defines what can be considered as personal information, which actors can operate with it and in what interactions they can be involved, etc.

A PCM is often made of *Privacy Conceptual Units (PCUs)*. For instance, some models [12] specify privacy into a list of privacy principles (fundamental, primary, or general guiding rules for the implementation of privacy protections), others [13] as a set of privacy harms (problems that a data subject may suffer as a consequence of an activity), yet some others [14] describe it as a set of technical goals (properties of the system-to-be). Note how this partition of the concept of privacy into conceptual units is not compulsory, e.g. some theories [15] conceive privacy without resorting to such partition, —yet all respond to some given conceptual model.

B. Privacy Normative Framework (PNF) and its components

A *Privacy Normative Framework* (PNF) provides *normative requirements* to be applied by all the methods claiming to abide by it, and it may include binding regulations as well as non-binding, recommended best practices. A PNF is composed of three types of prescriptive elements:

1. *Existential Constraints* which require (or preclude) the existence of specific elements in the methodology (specific *Tasks*, *Roles*, *Work Products*, etc.).
2. *Temporal Constraints* [3] on the elements of a methodology. They are expressed as entry or exit conditions on *Actions*, which must hold at a certain point in time (e.g. setting that an *Action* cannot be executed unless the condition is met) —besides they also affect indirectly the *Work Products* or *Work Units* from the method.
3. *Privacy Endeavor Requirements (PER)* are set on the system being developed in an endeavor. While both *Existential* and *Temporal Constraints* apply to the elements of the method itself, these *Requirements* apply to the products created when the method is enacted. Although a *Requirement Set* is typically considered one of the *Work Products* produced during an endeavor, in this case we are dealing with high-level requirements, which are provided by an external *Resource*, to be obeyed 'as is' by the system.

For example, the EU General Data Protection Regulation (GDPR) [16] provides a PNF that requires the existence (*Existential Constraint*) of a Data Protection Officer (DPO) *Role* with specifically allocated *Tasks*, prescribes that any data-processing-related *Task* cannot be performed unless an impact assessment *Process* has been carried out before (*Temporal Constraint*), and mandates a set of *Requirements* to be met by any system dealing with personal information. Note how the specific PNF defined by GDPR commits as well to a given PCM (viz. a set of principles relating to processing of personal data and a set of rights of the data subject), but these are different *Resources* even if referenced by the same source.

C. Privacy Engineering Code (PEC)

A *Privacy Engineering Code* (PEC) refines or clarifies the application of the PNF *under specific situations* or contexts. The PEC (sometimes known as 'code of conduct' or 'code of practice') includes a set of *Guidelines* that document how a *Constraint* or *Requirement* from the PNF can be applied whenever a methodology is enacted on a specific context or situation. The PEC may be typically subject to compliance or audit checks. For example, Art. 40 of the EU GDPR encourages that different institutions draw up codes of conduct "*...intended to contribute to the proper application of this Regulation, taking account of the specific features of the various processing sectors and the specific needs of micro, small and medium-sized enterprises.*"

D. Privacy Knowledge Base (PKB)

A *Privacy Knowledge Base* (PKB) is a piece of generally recognized knowledge that can be reused 'as is' in privacy-

[3] These are simply called Constraints by SEMDM, but we qualify them as Temporal to distinguish them from Existential Constraints.

engineering endeavors, and whose value and usefulness are collectively accepted. In our metamodel, a PKB is described as a set of *Model Units* (instances of some kind of element defined in a formal or conceptual *Language*), but which are provided by a methodology as a *Resource* rather than created by each endeavor. An analogy could be the set of standard libraries provided by most programming languages alongside the language specification itself, which define already developed software components to be integrated with others developed within an endeavor. A PKB can be used by different types of *Work Units* defined in the methodology.

Although privacy engineering is a nascent field, it has already developed certain amount of generally recognized knowledge, which is gathered in PKBs. For instance, privacy patterns [17] provide documented design solutions to common privacy problems in particular contexts. Privacy patterns can be described according to community-agreed templates and pattern languages which define the relations among them, and gathered together in privacy pattern repositories to be reused by privacy engineers. Some other examples of currently available PKBs are privacy design strategies [18] and privacy threats [19].

E. Privacy Engineering Roles (PER)

Some *Privacy Engineering Role* (PER) participates in one way or another in most privacy engineering methods. It represents someone who understands the privacy framework, is aware of the privacy engineering methodology elements that lead to the development of privacy-enhanced systems, and is able to apply them within the endeavor at hand. As such, *Privacy Engineering Roles* are characterized by their multidisciplinarity, being savvy in the three of privacy, engineering and the domain of the specific endeavor. As stated by Cranor [20], "*[a] privacy engineer is someone who understands the engineering and the privacy sides and works out strategies that allow people to protect privacy without getting in the way of building cool things.*"

Note that the *PER* represents an abstract role which shall be instantiated by more specific *Roles* defined by particular privacy engineering methods (e.g. *Privacy Requirements Engineer*, etc.), with the specific responsibilities set by the methodology. For instance, the Carnegie Mellon University M.Sc. In IT - Privacy Engineering enumerates a wide range of responsibilities that may be assumed by privacy engineers [21]: "*[…]develop technical solutions to help mitigate privacy vulnerabilities; analyze software designs and implementations from a privacy and UX perspective; research, document, and help remediate design decisions, operating procedures, or processes that may directly or indirectly contribute to future privacy risks; create cutting-edge privacy feature prototypes; help to lead better on privacy by example; and partner with key business, technical and legal stakeholders across various business groups to implement Privacy by Design.*"

Besides these *Privacy Engineering Roles*, any privacy engineering method may define additional, concrete roles (and their associated responsibilities) that must be considered at the methodology level. For instance, the EU GDPR identifies the roles of: data protection officer (DPO) with responsibilities in the privacy impact assessment, certification body with responsibilities in the audit and/or certification process, or independent supervisory authorities with responsibilities e.g. in the consultation prior to the processing. These roles need not represent privacy engineers (as they do not meet the aforesaid threefold savvy), yet they are part of the privacy engineering method (as they are involved in some of the tasks there defined).

IV. DESCRIPTION OF LINDDUN IN TERMS OF THE PRIVACY ENGINEERING METAMODEL.

For the proposed Privacy Engineering Metamodel to be useful, the concepts involved in privacy engineering methodologies should be mappable to either SEMDM methodology elements or to the extensions that we have introduced above. In particular, as a validation of the applicability of our metamodel, we have applied it to describe the LINDDUN methodology (a well-known privacy-engineering method) and its main elements, as shown next.

LINDDUN [4] is a model- and knowledge-based privacy engineering methodology aimed at systematically identifying the privacy threats in a system and the solutions that mitigate them, by following six linear steps, namely:

1. Define a data flow diagram (DFD), departing from either the requirements specification or the system architecture, while focusing on the internal data stores and the data flows that cross the organization boundaries, rather than on the internal processes.
2. Map privacy threat categories to DFD elements (just defined in the step above), according to a predefined table that details potential threat categories for each type of DFD element; while optionally discarding less likely threats, and reducing threats with common elements to a single one.
3. Identify threat scenarios, according to the guidance provided by a set of privacy-threat-tree patterns, describing threats in terms of misuse cases, and documenting any assumptions made.
4. Prioritize threats, depending on the risk associated to each one, according to the results of a risk assessment external to this methodology.
5. Elicit mitigation strategies, according to a taxonomy of strategies and a table that maps threat types to strategies.
6. Select Privacy-Enhancing Technologies (PETs), constrained by the mitigation strategies just elicited.

It is not difficult to realize how LINDDUN methodology can be modelled in terms of the elements of our Privacy Engineering Metamodel (including both native SEMDM elements and the extensions we have defined).

Each of the LINDDUN steps specifies what must be done in order to follow the methodology, that is, they define different types of *Tasks*. Besides, most of these steps also detail specific procedures to be followed in order to complete the respective task, that is, they also define some associated *Techniques*. These techniques are sometimes mandatory (e.g.

when threats are elicited, they must be refined using threat tree patterns, documented according to a threat description template together with any assumptions made), other times they are merely recommended (e.g. strategies and solutions should be respectively elicited using LINDDUN-provided mappings, but these are a mere convenience), and others are optional (e.g. DFDs can be created following specific techniques departing from specifications or architecture, but other techniques can be followed as long as the resulting DFD accurately models the data flows in the system). In some cases, LINDDUN does not even provide any technique for the respective task, but refers the reader to external sources (e.g. threat prioritization depends on applying techniques specified elsewhere, in order to compute the likelihood and impact of privacy threats). The said *Tasks* are also grouped into two *Processes*, namely the elicitation of privacy threats (which covers the first three steps) and the selection of mitigating solutions (the three last). These same *Processes* shall be iterated throughout the development cycle. And from the temporal perspective, these LINDDUN *Processes* can be performed during different *Phases* of a software and systems development methodology. Although the specific phases shall depend on and align with the development methodology employed, LINDDUN authors themselves suggest that these *Processes* can be applied several times during the "requirements" (i.e. inception) phase, during the "architecture" (i.e. elaboration) phase, or during the maintenance phase (on existing systems).

These *Work Units* (*Tasks*, *Techniques and Processes*) produce tangible results, i.e. different types of *Work Products*. More specifically, the DFD is a type of *Model* (hence the description of LINDDUN as "model-based"), whose *Model Units* (viz. external entities, data stores, data flows, and processes) respond to a *Language* and are represented (depicted) using a graphical *Notation*. Likewise, Misuse Cases [22] are types of *Models* employed to describe threats. And different types of *Documents* (the threat mapping table; and the lists of threats, assumptions, mitigation strategies and PETs) are created by instantiating the respective *Templates.* Besides, *Work Products* can be not only created, but also modified or merely read, e.g. when the outputs of a *Work Unit* are then used as inputs by another one. LINDDUN even allows for using *Work Products* that have been created elsewhere (e.g. the requirements specification or the architecture from which the DFD can be derived).

Some of the methodology steps are further supported by predefined catalogues of privacy threat trees, mitigation strategies and privacy-enhancing solutions. That is, the methodology provides three *Privacy Knowledge Bases* (once again, hence the "knowledge-based" feature of the methodology). These knowledge bases include each a list of atomic components or *Model Units* (threats, strategies, solutions), besides defining the relationships among them. These PKBs can be used by any *Producer* that applies the method, in order to simplify the elicitation processes by directly including these *Model Units* as needed, rather than coming up ex novo with other threats, strategies and solutions.

Table 1 (below) models the concepts defined by LINDDUN in terms of these method elements (*Processes, Tasks*, *Techniques*, *Work Products* and *Resources*) and provides the relations between them.

All LINDDUN's methodology elements are influenced by its underlying *Privacy Conceptual Model*, which consists of nine privacy properties (that is, *Privacy Conceptual Units*), viz. unlinkability, anonymity, pseudonimity, plausible deniability, undetectability, unobservability, confidentiality, awareness, and compliance. LINDDUN Privacy Threats are accordingly classified into seven categories (after whose initials LINDDUN is named), depending on the property respectively compromised: Linkability, Identifiability, Non-repudiation, Detectability, Disclosure of information, user Unawareness, and Non-compliance. It is through these categories that the influence of the *PCM* pervades the LINDDUN methodology. Thus, the structure of several *Templates* of *Documents* and *Privacy Knowledge Bases* in LINDDUN matches these

TABLE I. METHOD ELEMENTS IN LINDDUN

Processes	**Tasks**	**Techniques**		**Work Products**		**Resources**
	Task Kind	***Recom. Usage*** [a]	***Technique Kind***	***Action Type*** [b]	***Work Product Kind*** [c]	***Privacy Knowledge Base***
Elicitation of privacy threats	Define data flow	R	Create DFD from requirements	R	*System Requirements Specification*	
				C	Data Flow Diagram	
		R	Create DFD from Architecture	R	*System Architecture Document*	
				C	Data Flow Diagram	
	Map threats to data flow elements	M	Use threat mapping template	R	Data Flow Diagram	
				C	Threat Mapping table	
		O	Discard less likely threats	M	Threat Mapping Table	
		O	Combine threats ('reduction')	M	Threat Mapping Table	
	Elicit privacy threats	M	Refine threats	R	Threat Mapping Table	
				C	Threat List	Privacy Threat Tree Catalogue
		M	Document assumptions	C	Assumption List	
				M	Threat List	
		M	Document threats	R	Threat List	
				C	Misuse Cases	
Selection of mitigating solutions	Prioritize threats	–	–	R	*Risk Assessment Document*	
				M	Threat List	
	Elicit mitigation strategies	R	Map threats to strategies	R	Threat List	Mitigation Strategies (Taxonomy & Mapping)
				C	Mitigation Strategies List	
	Select Privacy-Enhancing Techniques	R	Map strategies to solutions	R	Mitigation Strategies List	Privacy-Enhancing Solutions Catalogue
				C	PETs List	

a. Recommended Usage: M = Mandatory, R = Recommended , O = Optional, D = Discouraged, F = Forbidden

b. Action Type: C= Create, M = Modify, D = Delete, R = Read-only

c. Work Products in italics are not defined by LINDDUN itself but elsewhere (nonetheless, they are used by LINDDUN).

categories, which guide as well the threat elicitation *Process* that yields the Threat List. The latter is employed to elicit the mitigation strategies, which in turn guide the selection of PETs, hence both are also indirectly affected by the threat categories.

V. Method Reusability and Integration through the Privacy Engineering Metamodel

It shall be noted that our *Privacy Engineering Metamodel* does not prescribe that all privacy methodologies incorporate all the types of *Resources* and other elements herein presented. Rather, we merely describe those elements which appear frequently in privacy methods, so as to offer a common theoretical model. What is more, as we discuss in this section, and siding with the objectives pledged by method engineering, the definition of different methodologies in compatible terms might ease the integration of elements or fragments from different privacy methods (whether in whole or in part) with one another and within generic (i.e. non-privacy-specific) software and systems engineering methodologies. Hence a specific privacy method may lack some of the metamodel elements; however, this should not be considered a drawback, but rather a feature, which matches the paradigm of method fragment reusability.

We anticipate that different privacy engineering methods will coexist, which respectively suit better the needs of specific fields, organizations, endeavors, legislations or technologies (i.e. there will not be any overarching method that covers the whole of privacy engineering). We assume that engineers may be willing to leverage these methods, so as to benefit from their usage beyond their initially planned scope, and integrating elements picked from different methods. And we posit that the definition of privacy methods in terms of the metamodel herein presented may ease their reuse and integration.

Indeed, the declared purpose of method engineering [6] consists in enabling the assembly of methodologies from method fragments coming from different sources, so as to develop new methods that fit better the endeavor at hand. SEMDM facilitates this methodological flexibility regarding each specific situation, by introducing three distinct layers for the metamodel, the methodology and the endeavor. Thus, two methods might be more easily integrated as long as they are described in compatible terms i.e. drawn from the same metamodel (to avoid an apples-and-oranges situation). Yet the metamodel not only provides a shared terminology, but also defines a series of hooks or extension points where elements from both methods can interface with each other. Then, departing from existent method parts or elements (e.g. various definitions of tasks, techniques, products), a methodologist can design new methods through different strategies (e.g. by assembling different fragments already available), always taking into account the goals that the method under construction is expected to fulfill. Therefore, method engineering responds to the question of how a method is developed in a context where relevance must be given to specific domain constraints and goals that the method under construction must fulfill (in our case, privacy constraints and goals). The answer comes by tailoring the method (rather than sticking to a predefined methodology set in stone) and selecting appropriate method elements to the specific context of each endeavor.

For instance, and keeping at the LINDDUN example, it does not prescribe any *Privacy Normative Framework*, nor does it feature any *Privacy Engineering Code*. A methodologist might anyway integrate LINDDUN with a specific *PNF* or *PEC,* by introducing the respective *Constraints, Requirements* and *Guidelines* as new method elements, and evaluating their impact in LINDDUN-defined elements. Likewise, it happens that LINDDUN is only focused on *Work Units* and *Work Products*, but it is agnostic regarding the *Producers* that perform and act upon them. Nonetheless, it should not be difficult to map elements from other methodologies (e.g. analysts, architects, etc.) to *Producer* 'placeholders'.

Furthermore, the *Tasks* defined by LINDDUN may depend on the integration with external methods. For instance, LINDDUN prescribes a *Task* to prioritize threats, which depends on the result of a risk assessment *Process*. However, neither does it tell what *Technique* to employ for that risk assessment, nor does it define the *Document Template* for the risk assessment document, leaving both to the implementer's decision. Other sources are available that provide techniques to compute a privacy threat likelihood and its impact which can be easily introduced, as long as they produce a list of risk indices for each threat which can be used for prioritization.

Moreover, LINDDUN does not even encompass all the possible privacy-related *Tasks* either. It only deals with some of them, while leaving out of its scope the creation of privacy policies; privacy testing, assessment, and auditing, etc., which could be specified by other methodologies. This can be eased by mapping LINDDUN *Tasks* onto different *Processes* from generic development methodologies (e.g. system analysis, risk assessment, architecture engineering), and completing the absent *Processes* with *Tasks* from other privacy-engineering methods.

All in all, LINDDUN elements would need to be integrated with elements from other methodologies in order to cover the whole development lifecycle, and they should be embedded within a mainstream development methodology (e.g. Agile, Unified Process, etc.) where privacy aspects would only play a limited part. All the intervening methods should be first modelled in terms of SEMDM so as to allow connecting their elements with one another. And only then, the resulting, composed methodology might be applied to new projects.

VI. Conclusion

It will be difficult that privacy engineering succeeds unless there is a common, shared understanding of its underlying concepts and the relationships between one another. We have defined such a common conceptual framework: a *Privacy Engineering Metamodel* that extends the SEMDM metamodel, and which paves the way to reuse and assemble methodologies. We have demonstrated the application of our metamodel by decomposing LINDDUN into its constituent elements, defined in terms of the SEMDM metamodel plus our extensions, and suggesting how these elements might be reused and integrated with other methodological approaches.

Most relevant concepts for privacy engineering can be mapped onto the elements provided by SEMDM, supplemented with our extensions. It should be noted that we do not claim either the novelty of these additional concepts (some of which have long been established in the general discipline of Engineering), or its exclusivity to the field of privacy engineering (as most can be applied to other categories of non-functional requirements as well); but just that we have identified them as appropriate to model privacy engineering, though they were not available straightaway from SEMDM. In fact, these extensions might well be considered as relevant contributions in the context of SEMDM itself.

Despite the proven need, a potential problem of our metamodel is the lack of a guarantee of adoption. It may happen that some proponents of privacy engineering methodologies refrain from recognizing other alternatives deemed to be as valid as theirs. We understand that, even if they cannot be reconciled, at least they may still agree on a common conceptual metamodel. We also plan to refine our metamodel by providing more examples of specific *Work Units*, *Producers* and *Work Products*, aligned to the taxonomies usually employed in the field of method engineering (e.g. OPFRO), which demonstrate its applicability to further privacy engineering methodologies.

It may also be the case that our proposed metamodel is only adopted within reduced academic circles, and contributes to further fragmentation rather than preventing it. That is why we aim to submit it to active standardization efforts. It is the case that the ISO and the IEC have taken the same challenge as well, by recently approving the work item (i.e. launching the development of a standard) ISO/IEC AWI 27550 on Privacy Engineering. This standard will aim to provide guidelines on how to engineer privacy in information systems considering different domains and under different development processes. We aim at contributing our metamodel to this standard, whose creation demonstrates the need to tackle the gap we are dealing with. Making it a *de jure* standard will foster its *de facto* adoption.

The metamodel we have presented is the result from extracting common features from several methodologies we are acquainted with. Here we have presented a specific validation case that exemplifies it (LINDDUN), but it may be well applied to other methodologies. Not only the existence of a shared conceptual model is itself key to foster the advance of the discipline; but also its modularity allows adapting it to specific endeavors or constraints. Rather than aspiring to come up with a single catch-all privacy engineering methodology; the modularity allows method engineers to create their own methodologies. We conjecture that ultimately, if the approach we propose succeeds, it will enable the integration of privacy engineering methodological elements with one another and within mainstream software and system development methodologies, and ultimately, will improve the privacy of the products or projects developed according to methodologies based on our metamodel —which is something yet to be proved in practice, so as to demonstrate the utility and effectiveness of our approach. In any case, we expect other members from the community to discuss the metamodel and enrich it with their own contributions.

It is the case, however, that the rigorous application of method engineering principles has been in general limited (e.g. to critical systems), and it remains a tool to organize the knowledge rather than a practical way to integrate different methodologies. Anyway, achieving the same result in the field of privacy engineering would still be a success nonetheless.

REFERENCES

[1] D. K. Mulligan, C. Koopman, and N. Doty, “Privacy is an essentially contested concept: a multi-dimensional analytic for mapping privacy: Table 1.,” *Philos. Trans. R. Soc. A Math. Eng. Sci.*, vol. 374, no. 2083, p. 20160118, 2016.

[2] S. Gurses and J. M. del Alamo, “Privacy Engineering: Shaping an Emerging Field of Research and Practice,” *IEEE Secur. Priv.*, vol. 14, no. 2, pp. 40–46, Mar. 2016.

[3] A. Cavoukian, “Privacy by Design The 7 Foundational Principles,” Toronto, Ontario (Canada), 2009.

[4] K. Wuyts and W. Joosen, “LINDDUN privacy threat modeling: a tutorial.” Department of Computer Science, KU Leuven, 2015.

[5] S. Brinkkemper, “Method Engineering: Engineering of Information Systems Development Methods and Tools,” *Inf. Softw. Technol.*, vol. 38, no. 4, pp. 275–280, 1996.

[6] B. Henderson-Sellers and J. Ralyté, “Situational Method Engineering: State-of-the-Art Review.,” *J. Univers. Comput. Sci.*, vol. 16, no. 3, pp. 424–478, 2010.

[7] C. Gonzalez-Perez and B. Henderson-Sellers, “A powertype-based metamodelling framework,” *Softw. Syst. Model.*, vol. 5, no. 1, pp. 72–90, 2006.

[8] ISO/IEC JTC 1/SC 27, “ISO/IEC 24744:2014. Software engineering -- Metamodel for development methodologies,” Geneva (CH), 2014.

[9] B. Henderson-Sellers and C. Gonzalez-Perez, “On the ease of extending a powertype-based methodology metamodel,” in *Meta modelling and ontologies : proceedings of the 2nd International Workshop on Meta-Modelling, WoMM 2006, October 12-13, 2006 in Karlsruhe, Germany*, 2006, pp. 11–25.

[10] D. J. Solove, “Conceptualizing privacy,” *California Law Review*, vol. 90, no. 4. pp. 1087–1155, 2002.

[11] H. Nissenbaum, “Privacy as contextual integrity,” *Wash. L. Rev.*, pp. 101–139, 2004.

[12] ISO/IEC JTC 1/SC 27, “Information technology -- Security techniques -- Privacy framework ISO/IEC 29100:2011,” Geneva (CH), 2011.

[13] D. J. Solove, “A Taxonomy of Privacy,” *Univ. PA. Law Rev.*, vol. 154, no. 3, pp. 477–560, 2006.

[14] A. Pfitzmann and M. Hansen, “A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management,” *Tech. Univ. Dresden*, pp. 1–98, 2010.

[15] S. Petronio, *Boundaries of Privacy: Dialectics of Disclosure*. New York, New York, USA: State University of New York Press, 2002.

[16] *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Da*. European Union: Official Journal of the European Union, L 119, pp. 1–88.

[17] “privacypatterns.eu - collecting patterns for better privacy.” [Online]. Available: https://privacypatterns.eu/. [Accessed: 19-Feb-2017].

[18] J.-H. Hoepman, “Privacy Design Strategies,” in *ICT Systems Security and Privacy Protection SE - 38*, vol. 428, N. Cuppens-Boulahia, F. Cuppens, S. Jajodia, A. Abou El Kalam, and T. Sans, Eds. Springer Berlin Heidelberg, 2014, pp. 446–459.

[19] K. Wuyts, R. Scandariato, and W. Joosen, “LIND(D)UN privacy threat tree catalog.” Department of Computer Science, KU Leuven, 2014.

[20] E. Heil, “Q&A: Privacy engineers could hold the key,” *TribLIVE.com*, 2012.

[21] “Privacy Engineering Careers-MSIT-Privacy Engineering - Carnegie Mellon University.” [Online]. Available: http://privacy.cs.cmu.edu/careers/. [Accessed: 19-Feb-2017].

[22] G. Sindre and A. L. Opdahl, “Templates for Misuse Case Description,” *7th Int. Work. Requir. Eng. Found. Softw. Qual. REFSQ 2001*, vol. 6, pp. 125–136, 2001.