This is a Non-Standards Track Work Product. The patent provisions of the OASIS IPR Policy do not apply.

Annex Guide to Privacy by Design Documentation for Software Engineers Version 1.0

Committee Note Draft 01

25 June 2014

Specification URIs

This version:

http://docs.oasis-open.org/pbd-se/pbd-se-annex/v1.0/cnd01/pbd-se-annex-v1.0-cnd01.doc (Authoritative)

http://docs.oasis-open.org/pbd-se/pbd-se-annex/v1.0/cnd01/pbd-se-annex-v1.0-cnd01.html

http://docs.oasis-open.org/pbd-se/pbd-se-annex/v1.0/cnd01/pbd-se-annex-v1.0-cnd01.pdf

Previous version:

N/A

Latest version:

http://docs.oasis-open.org/pbd-se/pbd-se-annex/v1.0/pbd-se-annex-v1.0.doc (Authoritative)

http://docs.oasis-open.org/pbd-se/pbd-se-annex/v1.0/pbd-se-annex-v1.0.html

http://docs.oasis-open.org/pbd-se/pbd-se-annex/v1.0/pbd-se-annex-v1.0.pdf

Technical Committee:

OASIS Privacy by Design Documentation for Software Engineers (PbD-SE) TC

Chairs:

Ann Cavoukian (commissioner.ipc@ipc.on.ca), Canada Office of the Information & Privacy Commissioner of Ontario

Dawn Jutla (dawn.jutla@gmail.com), Saint Mary's University

Editors:

Ann Cavoukian (commissioner.ipc@ipc.on.ca), Canada Office of the Information & Privacy Commissioner of Ontario

Fred Carter (fred.carter@ipc.on.ca), Canada Office of the Information & Privacy Commissioner of Ontario

Dawn Jutla (dawn.jutla@gmail.com), Saint Mary's University

John Sabo (john.annapolis@verizon.net), Individual

Frank Dawson (frank.dawson@nokia.com), Nokia Corporation
Sander Fieten, sander@fieten-it.com, Individual
Jonathan Fox (Jonathon_Fox@McAfee.com), Intel Corporation
Tom Finneran (tfinneran@gmail.com), Individual

Related work:

This document is related to:

- *Privacy by Design Documentation for Software Engineers Version 1.0*. Edited by Ann Cavoukian, Fred Carter, Dawn Jutla, John Sabo, Frank Dawson, Jonathan Fox, Tom Finneran, and Sander Fieten. Latest version: http://docs.oasis-open.org/pbd-se/pbd-se/v1.0/pbd-se-v1.0.html.

Abstract:

This Annex Guide serves as a primer for *Privacy by Design Documentation for Software Engineers (PbD-SE) Version 1.0*.

Status:

This document was last revised or approved by the OASIS Privacy by Design Documentation for Software Engineers (PbD-SE) TC on the above date. The level of approval is also listed above. Check the "Latest version" location noted above for possible later revisions of this document. Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at https://www.oasis-open.org/committees/pbd-se/.

Citation format:

When referencing this document the following citation format should be used:

[pbd-se-annex-v1.0]

Annex Guide to Privacy by Design Documentation for Software Engineers Version 1.0. Edited by Ann Cavoukian, Fred Carter, Dawn Jutla, John Sabo, Frank Dawson, Sander Fieten, Jonathan Fox, and Tom Finneran. 25 June 2014. OASIS Committee Note Draft 01. http://docs.oasis-open.org/pbd-se/pbd-se-annex/v1.0/cnd01/pbd-se-annex-v1.0-cnd01.html. Latest version: http://docs.oasis-open.org/pbd-se/pbd-se-annex/v1.0/pbd-se-annex-v1.0.html.

works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Table of Contents

1 Introduction

This annex describes a methodology to help engineers to model and document *Privacy by Design* (PbD) requirements, translate the principles to conformance requirements within software engineering tasks, and produce artifacts as evidence of PbD-principle compliance.

This Annex provides:

- An elaborated expression and explanation of the Privacy by Design principles in the context of software engineering. In effect, it closes a communications, requirements, and operations gap among policymakers, business stakeholders, and software engineers.
- Elaboration of the mapping of the Privacy by Design principles to engineering-related sub-principles, and to documentation, and thus PbD-SE compliance criteria.
- Elaboration of privacy considerations for the entire software development life cycle from software conception to software retirement.
- Intermediate Software engineering Documentation Checklists
- A process to ensure that privacy requirements are considered throughout the entire software development life cycle from software conception to software retirement.
- A methodology for an organization and its software engineers to produce and reference privacy-embedded documentation to demonstrate compliance to Privacy by Design principles.
- A Privacy Use Template that helps software engineers document privacy requirements and integrate them with core functional requirements.
- An example Privacy by Design Reference Architecture for software engineers to customize to their context, and Privacy Properties that software solutions should exhibit.
- Privacy by Design Patterns (developed in a future version of Annex)
- Privacy by Design for Maintenance and Retirement (developed in a future version of Annex)

This specification is targeted to all software engineers. They may work in (virtual) organizations and/or on projects of all sizes. This includes software engineers working in platform teams or on solo platforms.

Software engineers are responsible for implementing, and documenting or referencing documentation to show compliance to *Privacy by Design* principles. However, as software engineers operate in larger contexts, this specification is also of interest and use to their project managers, business managers and executives, privacy policy makers and compliance managers, privacy and security consultants, auditors, regulators, and other designers and users of systems that collect, store, process, use, share, transport across borders, exchange, secure, retain or destroy personal data. In larger organizations, where subject matter experts and organizational

stakeholders have clear roles in the SDLC, their contributions may be an explicit part of the documentation..

1.1 Non-Normative References

[BIRO 2009]

The BIRO Project, Cross-border flow of health information: is 'privacy by design' enough? Privacy performance assessment in EUBIROD. Available at http://www.ncbi.nlm.nih.gov/pubmed/22562711.

[Cavoukian 1995]

Privacy-Enhancing Technologies: The Path to Anonymity, Volume II, available at http://www.privacybydesign.ca/content/uploads/1995/03/anoni-v2.pdf

[Cavoukian 2011]

Privacy by Design: The 7 Foundational Principles Implementation and Mapping of Fair Information Practices at www.ipc.on.ca/images/Resources/pbd-implement-7found-principles.pdf

[Cavoukian 2012]

Privacy by Design: Leadership, Methods, and Results, Chapter 8: http://link.springer.com/chapter/10.1007%2F978-94-007-5170-5_8, 5th Int. Conference on Computers, Privacy & Data Protection European Data Protection: Coming of Age, Springer, Brussels, Belgium.

[CICA 2014]

CICA/AICPA Privacy Maturity Model, available at http://www.cica.ca/resources-and-member-benefits/privacy-resources-for-firms-and-organizations/docs/item48094.pdf

[Dennedy et al 2014]

Michelle Finneran Dennedy, Jonathan Fox, Thomas Finneran (2014). The Privacy Engineer's Manifesto: Getting from Policy to Code to QA and Value, Apress, Jan. 2014, 400 pages.

[Jacka and Keller 2009]

Mike Jacka, Paulette Keller. Business Process Mapping: Improving Customer Satisfaction. John Wiley and Sons. p. 257. ISBN 0-470-44458-4.

[Jutla and Bodorik 2005]

Dawn N. Jutla, Peter Bodorik (2005), Sociotechnical Architecture for Online Privacy, IEEE Security and Privacy, vol. 3, no. 2, pp. 29-39, March-April 2005, doi:10.1109/MSP.2005.50.

[Jutla et al 2013]

Dawn N. Jutla, Peter Bodorik, Sohail Ali (2013). Engineering Privacy for Big Data Apps with the Unified Modeling Language. IEEE Big Data Congress 2013: 38-45. Santa Clara.

[Jutla 2014]

Dawn N. Jutla, Evolving OASIS Privacy by Design Standards, April 9, 2014, available at http://csrc.nist.gov/news_events/privacy_workshop_040914/nist_pew_2014dawn_jutla.pdf

[Jutla 2014a]

Dawn N Jutla, M2M Vehicle Telematics and the OASIS Privacy by Design Documentation for Software Engineers (PbD-SE), European Identity and Cloud Conference, Munich, April 15 2014, slide deck available to attendees.

[Jutla 2014b]

Dawn N. Jutla, Overview of the OASIS PbD-SE and PMRM emerging privacy standards, PRIPARE workshop, Cyber Security and Privacy Forum, Athens, May 22, 2014.

[Jutla et al 2014]

Dawn N. Jutla, Ann Cavoukian, John T. Sabo, Michael Willett, Fred Carter, Michelle Chibba, Operationalizing Privacy by Design Documentation for Software Engineers and Business: From Principles to Software Architecture, submitted for journal review, April 21, 2014.

[PbD-FIPPS]

Ann Cavoukian, The 7 Foundational Principles: Implementation and Mapping of Fair Information Practices, http://bit.ly/vjvrPE January 2011

[NIST 800-53]

Security and Privacy Controls for Federal Information Systems and Organizations Revision 4, Appendix J: Privacy Controls Catalog

[Shostack 2014]

Adam Shostack (2014), Threat Modeling: Designing for Security, Wiley, Feb 2014.624 pages.

[Zachmann 1987]

J. Zachmann. A framework for information systems architecture. IBM Systems Journal, Vol. 26. No. 3, 1987.

NOTE: The proper format for citation of technical work produced by an OASIS TC (whether Standards Track or Non-Standards Track) is:

[PbD-SE Annex 1.0]

Annex to the Privacy by Design Documentation for Software Engineers Version 1.0, Edited by Ann Cavoukian, Fred Carter, Dawn Jutla, John Sabo, Frank Dawson, Jonathan Fox, Tom Finneran, Sander Fieten, 25 June 2014.OASIS Committee Note 01. Principal URI http://docs.oasis-open.org/pbd-se/pbd-se-annex/v1.0/cnd01/pbd-se-annex-v1.0-cnd01.doc

The permanent "Latest version" URI will be:
http://docs.oasis-open.org/pbd-se/pbd-se-annex/v1.0/pbd-se-annex-v1.0.doc

2 *Privacy by Design* for Software Engineers

This section describes the default context of *Privacy by Design* and lays out the meaning of its principles in terms specific to software engineers. The *Privacy by Design* framework was unanimously recognized by international privacy and data protection authorities in October 2010 as an essential component of fundamental privacy protection; Privacy and data protection authorities resolved to encourage the adoption of *PbD* principles as guidance to establishing privacy as an organization's default mode of operation;

The *Privacy by Design* framework consists of seven high-level and interrelated principles that extend traditional Fair Information Practice Principles to prescribe the strongest possible level of privacy assurance. A mapping of PbD principles to the FIPPs is provided below.

Table 2.1: *Privacy by Design* Principles Mapped to Fair Information Practice Principles

PbD Principles	**Meta-FIPPs**	**Traditional FIPPs**
1. Proactive Not Reactive; Preventative Not Remedial	Leadership & Goal-Setting	---
2. Privacy as the Default Setting	Data Minimization	Purpose Specification Collection Limitation Use, Retention & Disclosure Limitation
3. Privacy Embedded into Design	Systematic Methods	---
4. Full Functionality – Positive-Sum, not Zero-Sum	Demonstrable Results	---
5. End-to-End Security Full Life-Cycle Protection	Safeguards	Safeguards
6. Visibility and Transparency - Keep it Open	Accountability (beyond data subject)	Accountability Openness Compliance
7. Respect for User Privacy – Keep it User-Centric	Individual Participation	Consent Accuracy Access Redress

Privacy by Design: *The 7 Foundational Principles Implementation and Mapping of Fair Information Practices* at www.ipc.on.ca/images/Resources/pbd-implement-7found-principles.pdf

As with traditional FIPPs, PbD principles set forth both substantive and procedural privacy requirements, and can be applied universally to information technologies, organizational systems and networked architectures. This specification prescribes the application of PbD principles to software engineering documentation.

This specification enables software engineers to embed privacy into the design and architecture of software-enabled data systems, in order to minimize data privacy risks without diminishing system functionality. The review seeks to aid the whole team and executive level to understand the PbD principles in a software engineering context.

2.1 Proactive not Reactive; Preventative not Remedial

This principle emphasizes early privacy risk mitigation methods, and requires a clear commitment, at the highest organizational levels, to set and enforce high standards of privacy – generally higher than the standards set by laws and regulation. This privacy commitment should be demonstrably shared throughout by relevant stakeholders (internal and external) in a culture of continuous improvement.

2.1.1 Demonstrable Leadership

Software engineering methods and procedures are in place to ensure a clear commitment, from the highest levels, to prescribe and enforce high standards of privacy protection, generally higher than prevailing legal requirements.

2.1.2 Defined Community of Practice

Software engineering methods and procedures are in place to ensure that a demonstrable privacy commitment is shared by organization members, user communities and relevant stakeholders.

2.1.3 Proactive and Iterative

Software engineering methods and procedures are in place to ensure continuous processes are in place to identify privacy and data protection risks arising from poor designs, practices and outcomes, and to mitigate unintended or negative privacy impacts in proactive and systematic ways.

2.2 Privacy as the Default

This principle emphasizes establishing firm, preferably automatic, limits to all collection, use, retention and disclosure of personal data in a given system. Where the need or use of personal

data is not clear, there is to be a presumption of privacy and the precautionary principle is to apply: the default choice should be the most privacy protective.

This *Privacy by Design* principle:

- has the greatest impact on managing data privacy risks, by effectively eliminating risk at the earliest stages of the data life cycle.
- prescribes the strongest level of data protection and is most closely associated with limiting use(s) of personal data to the intended, primary purpose(s) of collection; and
- is the most under threat in the current era of ubiquitous, granular and exponential data collection, uses, disclosures and retention.

The default starting point for designing all software-enabled information technologies and systems mandates NO collection of personally data —unless and until a specific and compelling purpose is defined.

As a rule, default system settings are maximally privacy-enhancing. This rule is sometimes described as “data minimization” or “precautionary” principle, and must be the first line of defense. Non-collection, non-retention and non-use of personal data is integral to, and supports, all of the other PbD principles.

2.2.1 Purpose Specificity

Privacy commitments are expressed by documenting clear and concise purpose(s) for collecting, using and disclosing personal data. Purposes may be described in other terms, such as goals, objectives, requirements, or functionalities. In the context of engineering software designs:

- Purposes must be limited and specific; and
- Purposes must be written as functional requirements.

2.2.2 Limiting Collection, Use, and Retention

The software should be designed in such a way that personal data is collected, used, disclosed and retained:

- in conformity with the specific, limited purposes;
- in agreement with the consent received from the data subject(s); and
- in compliance with legal requirements.

Consistent with data minimization principles, strict limits are in place in each phase of the data processing life cycle engaged by the software under development. This includes:

2.2.2.1 Limiting Collection

The software engineer ensures techniques, systems and procedures are put in place to:

1. specify essential versus optional personal data to fulfill identified purposes;
2. associate sensitivity levels with personal data collected

3. periodically review data requirements;
4. document individual consent to collect sensitive personal data;
5. monitor the collection of personal data to ensure it is limited to that necessary for the purposes identified, and that all optional data is identified as such;
6. link stated purpose of collection to the data source identification;
7. ensure auditability of legal or business adherence to collection limitation;
8. assign time expirations to data at time of collection or creation;
9. establish levels or types of identity such as gradations of non-identifiable, identifiable or identified data collection and processing that need to be supported; and
10. establish limits to collection associated with levels or types of data subject identity.

2.2.2.2 Collecting by Fair and Lawful Means

The software engineer ensures that techniques, systems, and procedures are put in place to

1. review and confirm for relevant methods, before they are implemented, that personal data is obtained
   (a) fairly, without intimidation or deception, and
   (b) lawfully, adhering to all relevant rules of law.
2. associate “fair and lawful” collection with the data source(s).

2.2.2.3 Collecting from Third Parties

The software engineer ensures that techniques, systems and procedures are put in place to:

1. ensure that personal data collection from sources other than the individual are reliable ones that also collect data fairly and lawfully. This requires that:
   a. due diligence be performed before establishing a relationship with a third-party data provider.
   b. privacy policies, collection methods, and types of consents obtained by third parties be reviewed before accepting personal data from third-party data sources.
2. document and, where necessary, seek consent where the software produces or acquires additional data about individuals.

NOTE: These requirements are specifically for personal data that is collected through a third party. The general requirements as documented in the above sections also apply.

2.2.2.4 Uses and Disclosures

The software engineer ensures techniques, systems and procedures are put in place to:

1. limit all uses and disclosures of personal data to the specified purposes (and for which the individual has provided implicit or explicit consent);
2. differentiate personal data by both type and quantity, and treat accordingly;
3. anticipate emergency and exceptional uses and disclosures;
4. assign and observe time expirations associated with uses;
5. tie future uses of personal data to the original collection purpose(s);
6. document whether selected “secondary” use(s) may be allowed under law;

7. secure individual consent, where necessary, for disclosures to third parties;
8. document justification(s) for all disclosures without subject consent;
9. inform third parties of relevant collection, use, disclosure and retention requirements, and ensure adherence;
10. audit retention limits and resulting destruction; and
11. ensure security of data transfers.

2.2.2.5 Retention

The software engineer ensures that techniques, systems and procedures are put in place to:

1. limit retention no longer than needed to fulfill the purposes (or as required by law or regulations) and thereafter appropriately dispose of such data;
2. document retention policies and disposal procedures;
3. retain, store, and dispose of archived and backup copies of records in accordance with applicable retention policies;
4. ensure personal data is not kept beyond the standard retention period unless a justified business or legal reason exists for doing so; and
5. consider contractual requirements when establishing retention practices that may be exceptions to normal policies/practices.

2.2.2.6 Disposal, Destruction and Redaction

The software engineer shall ensure techniques, systems and procedures are put in place to:

1. regularly and systematically destroy, erase, or de-identify personal data that is no longer required to fulfill the identified purposes;
2. dispose of original, archived, and backup records in accordance with the retention and destruction policies;
3. carry out disposal in a manner that prevents loss, theft, misuse, or unauthorized access;
4. document the disposal of personal data;
5. within the limits of technology, locate and remove or redact specified personal data about an individual as required; and
6. consider contractual requirements when establishing disposal, destruction, and redaction practices if these may result in exceptions to the normal policies/practices.

2.3 Privacy Embedded in Design

This principle emphasizes integrating privacy protections into the methods by which data systems are designed and developed, as well as how the resulting systems operate in practice. A systematic approach to embedding privacy is to be adopted —one that relies upon accepted standards and frameworks. Privacy impact and risk assessments shall be carried out, documenting the privacy risks and measures taken to mitigate those risks, including consideration of alternative design options and choice of metrics. The privacy impacts of the

resulting technology, operation or data architecture, and their uses, shall be demonstrably minimized, and not easily degraded through use, misconfiguration or error.

2.3.1 Holistic and Integrative

The software engineer ensures that privacy commitments are embedded in holistic and integrative ways by adopting as broad a scope as possible when identifying and mitigating privacy risks.

2.3.2 Systematic and Auditable

The software engineer ensures that a systematic, principled approach is adopted that relies upon accepted standards and process frameworks, and is amenable to external review.

2.3.3 Reviewed and Assessed

The software engineer ensures that detailed privacy impact and risk assessments are used as a basis for design decisions.

2.3.4 Human-Proof

The software engineer ensures that the privacy risks are demonstrably minimized and not increased through use, misconfiguration, or error.

2.4 Full Functionality — Positive-sum, Not Zero-sum

This principle seeks to accommodate all legitimate interests and objectives in a positive-sum "win-win" manner. When embedding privacy into a given technology, process, or system, it shall be done in such a way that functionality is not impaired, and to the greatest extent possible, that all requirements are optimized. All non-privacy interests and objectives must be clearly documented, desired functions articulated, metrics agreed upon and applied, and zero-sum trade-offs rejected wherever possible, in favour of solutions that enable multi-functionality and maximum privacy.

2.4.1 No Loss of Functionality

The software engineer ensures that embedding privacy does not impair functionality of a given technology, process or network architecture.

2.4.2 Accommodate Legitimate Objectives

The software engineer ensures that all interests and objectives are documented, desired functions articulated, metrics agreed, and trade-offs rejected in the first instance, when seeking a solution that enables multi-functionality

2.4.3 Practical and Demonstrable Results

The software engineer ensures that, wherever possible, optimized outcomes are published for others to emulate and to become best practice.

2.5 End to End Security – Lifecycle Protection

This principle emphasizes continuous protection of personal data across the entire domain in question, whether the personal data is at rest, in motion or in use from initial collection through to destruction. There shall be no gaps in either protection of, or accountability for personal data. Applied security standards are to assure the confidentiality, integrity and availability of personal data throughout its lifecycle including, among other things, appropriate use of encryption techniques, strong access controls, logging and auditing techniques, and methods of secure destruction.

2.5.1 Protect Continuously

The software engineer ensures that personal data is continuously protected across the entire system scope and throughout the data life-cycle, from creation to destruction.

2.5.2 Control Access

The software engineer ensures that access to personal data is commensurate with its degree of sensitivity, and is consistent with recognized standards and criteria.

2.5.3 Use Metrics and Satisfy Privacy Properties

The software engineer ensures that security standards are applied that assure the confidentiality, integrity and availability of personal data, and are amenable to verification. The software engineer ensures that solutions support user/data subject-level and system-level privacy properties (see Section 4) and are amenable to verification. The reduction of security and privacy risks should be quantified and reported regularly.

2.6 Visibility and Transparency – Keep it Open

The software engineer shall create the foundation for for accountable software by providing, to relevant stakeholders, appropriate information and evidence about how the software or system fulfills stated promises and objectives. Demonstrating visibility and transparency enhance understanding among software users, and provide for informed choices by users/data subjects. Robust visibility and transparency enhance the capacity for independent verification.

2.6.1 Open Collaboration

The software engineer ensures that privacy requirements, risks, implementation methods and outcomes are documented throughout the development lifecycle and communicated to project members and stakeholders.

2.6.2 Open to Review

The software engineer ensures that the design and operation of software systems demonstrably satisfy the strongest privacy laws, contracts, policies and norms (as required).

2.6.3 Open to Emulation

The software engineer ensures that the design and operation of privacy-enhanced information technologies and systems are open to scrutiny, praise and emulation by all.

2.7 Respect for User* Privacy – Keep it User-Centric

The software engineer shall keep the interests of the individual user uppermost by offering strong privacy defaults, appropriate notice, and user-centric and user-friendly interfaces. A key objective of this principle is to empower users/data subjects to play active roles in managing personal data through mechanisms designed to facilitate informed consent, direct access and control, and redress.

2.7.1 Anticipate and Inform

The software engineer ensures that the software is designed with user/data subject privacy interests in mind, and convey privacy properties (where relevant) in a timely, useful, and effective way.

2.7.2 Support User / Data Subject Input and Direction

The software engineer ensures that technologies, operations and networks allow users/data subjects to express privacy preferences and controls in a persistent and effective way.

2.7.3 Encourage Direct User / Data Subject Access

The software engineer ensures that software systems are designed to provide users/data subjects direct access to data held about them, and an account of uses and disclosures.

3 Operationalizing the PbD Principles in Software Engineering

This section defines a technology governance methodology for operationalizing PbD Principles through Software Engineering tasks. This methodology is agnostic to software development life cycle (SDLC) methodology or organizational structure. It is directed at ensuring the privacy engineering methodology is operationalized so that it is functionally sustained and not solely dependent on the good will and memory of a single individual. Each organization or entity using the methodology will need to right-size steps based on their resources, complexity and size.

The methodology emphasizes an expectation of iteration over one or more of its steps. Several steps address the questions: what's changed with personal data, and are privacy properties upheld. The methodology documents the processes, policies, standards, and guidelines that are being used to ensure privacy requirements are identified and incorporated and addressed in the development process and methodology. It includes pre-existing privacy risk models.

3.1 Organizational Privacy Readiness

In order to demonstrate adherence to Privacy by Design principles, an organization shall:

- Establish executive leadership and commitment to operationalizing Privacy by Design.
- Identify who in the engineering organization is responsible for Privacy by Design. Depending on size and structure of overall organization, this person may hardline or dotted-line report to the CPO or be the CPO.
- Determine these resources' responsibilities; i.e., is this person responsible for building the organization's overall privacy program or is limited to the engineering function.
- Identify who are candidates for privacy engineering leads for projects. This person may be both the privacy lead for the organization and for the project, or the tasks may be divided among several people, according to the extent of the firm's resources.
- Determine privacy resource's responsibilities across projects; i.e., is the person(s) also the privacy architect and engineer and responsible for QA or does he/she or they lead a team.
- Determine who within the organization is responsible and accountable for privacy within the components of the engineering process and their relationship to the privacy engineering lead for the organization and for the project.
- Determine training, communication and knowledge transfer/management mechanisms to ensure role is functionalized and not personality dependent.

Privacy capabilities maturity varies across size and age of organizations and industry. The output of this methodological step documents the working contexts of software engineers with respect to privacy readiness. Software engineers in medium to large organizations will not produce all the documentation required for this step. Others in their organizations will, but the software engineer must *reference* and show a working knowledge of the content of organizational privacy documentation, including privacy policies and privacy resources, in her/his organization.

Because, software engineers in small organizations may take on the responsibility of a CPO/privacy manager/privacy resource in order to comply with PbD principles, the guidance in this step to identify and create privacy-related roles, responsibilities, and accountabilities is paramount.

3.2 Scope and Document Privacy Requirements

Based on an analysis of the product (e.g., defined use cases or user/data subject stories), the software organization shall scope and document initial product privacy requirements. These requirements establish the default conditions for privacy within the product.

Stakeholders shall define and document the key user/data subject privacy stories, or privacy use cases, using the Privacy Use Template (section 5.1) or the more comprehensive OASIS PMRM methodology [2013] or EQUIVALENT, and users'/data subjects' privacy experience requirements that scope the products' privacy requirements or privacy features.

Software engineers shall model and classify data and behavior with common tools, e.g..one or more of spreadsheets, data flow, data models, UML sequence and/or activity diagrams, or equivalent diagrams to scope the integration of privacy user/data subject-level functionality and system properties with the software's functional requirements. These documents may be used in privacy threat analysis.

3.3 Conduct Privacy Risk Analysis and Privacy Property Analysis

For whatever is being engineered, the software organization shall examine the most recent previous risk assessment reports available. If they are not up-to-date, the organization shall produce a threat assessment report (including documenting threat models, e.g. [Shostack, 2014], privacy impact assessment, and business impact summary.

For whatever is being engineered, scoping and documenting privacy requirements (Step 2, Section 3.2) produces a privacy requirements report following the Privacy Use Template found in section 5 of this specification document

For whatever is being engineered, the software organization shall produce a privacy controls' evaluation and selection report. This report shall address how well the selected controls satisfy privacy properties (see Section 5.3.1). The evidence from this and previous steps are used to determine the level of privacy resourcing.

3.4 Identify Privacy Resource(s) to support the Solution Development Team

The identity of the team's privacy "champion(s)" that liaises with the responsible Privacy Officer within the organization (if one exists), and the product's privacy vision to set a goal for the privacy impact threshold for the product, shall be documented. There is an expectation of ongoing interaction between privacy officer/privacy engineer and software engineers for writing documentation that can be reviewed correctly. Note that a software engineer can take on accountable and responsible roles for privacy, if necessary.

Responsibilities of the privacy resource(s) include maintaining PbD Principles in work products/services. Privacy resource(s) would work with the data stewards and other parts of the team in determining the privacy impact of each data attribute. Stakeholders, including software engineers, work together to certify that deployed solutions comply with PbD principles.

[Context: Note that the state of the art shows that some software engineering teams are better resourced than others, regardless of levels of privacy threats, and that privacy resources are scarce as software organizations hold costs down. Thus this step may lead to different privacy resource allocation across teams.]

3.5 Assign Responsibility for PbD-SE Operationalization and Artifacts Output

The software organization shall:

1. Document who in the engineering organization is responsible for privacy engineering, who within the larger organization is responsible for privacy and software engineering, and the executive champion for privacy engineering
2. Document the team's privacy resource's responsibilities.
3. Document who is the privacy engineering lead for a specific project, and her/his responsibilities
4. Document the engineers responsible for privacy within the components of the engineering process.
5. Document who within the organization the privacy engineering lead or designate works with to address requirements for overall organizational readiness for deployment or release of privacy engineered solution.

This step documents the engineers' working environment with respect to privacy engineering readiness. Organizations may use a framework such as the RACI model [Jacka and Keller, 2009]--- responsibility, accountability, consulting/collaborative, informed – to document the assignment of various resources for operational PbD-SE and artifact production. This step achieves sharing of the responsibility for PbD principles across the solutions engineering team in a larger project management process. Responsibilities and associated metrics will be tracked throughout the work stream. The output of this step is the documentation of the accountable and responsible resources, as well as those resources that act in a collaborative/consulting manner, and those who simply stay informed.

3.6 Design

A privacy reference architecture may be created and documented as a basis for later software engineering of complementary software classes or components. Software engineers may use the privacy services-based (SOA) reference architecture, shown in Fig. 5.5 as a high-level guide. This services-oriented architecture (SOA) will be complemented with detailed and contextual architectural viewpoints of the eventual software product/solution consisting of privacy components, connectors, data repositories, and metadata. Design classes, mappings to

implementation classes, UI architecture and designs, and selection of technologies, shall also be documented output of this stage.

The resulting architecture shall satisfy system-level privacy properties, such as minimizing observability, identifiability and linkability with system use, traceability and auditability, and accountability. Furthermore, the architecture shall satisfy user/data subject-level properties, including comprehension, consciousness, choice, and consent around privacy, and support context, confinement (data minimization and proportionality), and consistency [Jutla and Bodorik, 2005, Jutla et al, 2014].

3.7 Review Code

Software engineers shall execute specific privacy tests, formulated early at the privacy requirements specification stage, to examine the privacy compliance issues. Privacy and security metrics around satisfaction of privacy properties (as per Section 3.6) shall guide evaluation. User/data subject testing and/or studies may also inform the outcome of this step. This step also ensures screening of third party code for privacy violations before incorporation in existing software.

3.8 Plan for Retirement of Software Product/Service/Solution

Product/maintenance teams shall create privacy ramp-down guidelines in a retirement plan.

A non-exhaustive list of examples of what such a plan may include, but is not limited to:

- If software is consumer facing, organizations communicate to consumers that services are shutting down, and may inform about archiving or disclosure/sharing requirements for databases.
- If the data controller changes, companies have an obligation to inform users/data subjects, and if applicable give choice as to deletion, before movement to a new data owner.
- If an organization uses data processors, or third-party service providers, similar communication to data processors are needed. Retirement plans should contain a quality statement around experience on ramp down.
- For software and data on hardware, security controls, e.g. NIST directives on hard disk erasure, are documented in the retirement plan.
- In registration countries, organizations notify Data Processing Authority (DPA) that the DBs are not active anymore.

3.9 Review Artifacts throughout the SDLC

The context of the methodology is through a data maintenance life cycle for a software-engineered product/service. Documents shall be completed but also reviewed periodically during this life cycle by different stakeholders. For example, the privacy legal team reviews the privacy document artifacts to ensure compliance to PbD principles.

3.10 Sign off with PbD-SE methodology check list

This step verifies that proper documentation exists. A checklist is useful for managers and responsible stakeholders to assess, at a glance, whether PbD principles are considered and privacy documentation generated and/or referenced.

Table 3.1 shows an example RACI chart that can act as a checklist. It is expected that organizations will have different RACI assignments according to their specific contexts (e.g. size and organization).

Table 3.1 RACI Chart for Software Engineers

PbD-SE Methodology Step	Documents to be referenced or produced	Software Eng-ineer	Privacy Res-ource	Proj-ect Man-age-ment.	Mana ge-ment.	Third Party	User	Data Sub-ject	Check -list item
3.1 Reference Organization -al Readiness	Privacy Policy Document	CI	RACI	CI	ACI	I	CI	I	✓
	Privacy Roles/Training Program in Organization	I	RACI	CI	AI	I			X
3.2 Scope Privacy Requirement s & Reference Architecture	Functional Privacy Require-ments, preliminary controls identification & hooks to Reference Architecture	RA	RACI	ACI	AI	RAI	CI		✓
3.3 Conduct Risk and Privacy Property Analysis	Traceability diagrams and other documentatio n to show consideration	CI	RACI	CI	AC	CI	-		✓

	of privacy properties								
	Risk analysis Final controls identification	CI	RACI	CI	ACI	CI	- -	CI	✓
3.4 Identify Privacy Resource Allocation	Privacy resource allocation to SE team	I	RACI	RI	AI	I	-		✓
3.5 Create RACI for Producing Artifacts	RACI assignment to artifact production	RCI	CI	RACI	AI	-	-		✓
3.6 Customize Privacy Architecture	Privacy Architecture (incl. services identification)	RA	ACI	A CI	AI	I	-		✓
3.7 Conduct Periodic Review	Review of Artifacts throughout the SDLC	RA	CI	RACI	AI	-	-	C	✓
3.8 Execute Code Testing & Privacy Evaluation	Testing and evaluation for satisfying privacy properties	RA	RCI	RA CI	AI	-	C	C	✓
3.9 Create Retirement Plan	Plan for retirement of software solution	CI	RACI	RACI	ACI	I	I	C	✓
3.10Sign-off	Sign off with checklist	RACI	RACI	RACI	AC	-	-		✓

4 Software Development Life Cycle Documentation for *Privacy by Design*

This section directly relates to normative clauses in Table 2.1 of the PbD-SE specification [PbD-SE 01]. Elaboration of privacy documentation, including tools and visual models for privacy are provided in this section. Tools and visual models help software engineers generate and document privacy requirements and design, and visualize and embed *Privacy by Design* requirements through encapsulated privacy services, components, or patterns in their product designs and implementations. The current version of this section specifies types of and equivalence of documentation to fulfill obligations, listed in Table 2.1 [PbD-SE 01], for a significant subset of *PbD* principles, particularly for *Privacy by Default* and *Privacy Embedded in Design*.

4.1 *Privacy by Design* Use Case Template for Privacy Requirements

This subsection describes the type of tools and techniques that software engineers employ for a comprehensive understanding and documentation of privacy in a software development project and operationalizing *Privacy by Design* into the requirements analysis phase of the software development life cycle. This specification is flexible in that it allows for use of EQUIVALENT types of tools, methods, or models to those described in this section in order to satisfy PbD-SE compliance.

Software engineers show consideration of privacy when they do the equivalent of the following: include user privacy stories or privacy use cases in their functional analysis and designs; follow privacy requirements elicitation methodologies, such as, the *Privacy by Design* Use Template (elaborated in [PMRM-01]) that expresses privacy requirements as functional requirements; and/or use pragmatic diagramming and documentation tools to visualize, record, and enact *Privacy by Design*.

Applying *Privacy by Design* to the software engineering discipline requires “operationalizing” PbD principles. Among other things, this operational focus requires breaking down abstract PbD principles, FIPPs, privacy policies and privacy related business processes into components. At times this decomposition process can be extremely complex. Using a standardized template can help to make this complexity manageable by providing a structure for analysis and exposing a comprehensive privacy picture associated with a specific use case or set of user stories.

Because documentation artifacts memorialize analysis and actions carried out by stakeholders, a Privacy Use Template can aid in their production. Additionally, adopting a Template throughout the organization and across organizations has multiple benefits:

- A standardized use template can reduce the time and cost of operationalizing PbD and improve the quality and reusability of documentation
- It provides all stakeholders associated with the specified software development project within an organization a common picture and a clearer understanding of all relevant privacy components of the project

- It can expose gaps where PbD analysis has not been carried out or where implementation has not been initiated or completed
- It is a tool to map privacy policies, requirements and control objectives to technical functionality
- A standardized template also facilitates the re-use of knowledge for new applications and the extension of Privacy by Design principles more broadly throughout an organization
- Finally, where code must bridge to external systems and applications, a standardized template will help ensure that Privacy by Design principles extend to the protection of personal data transferred across system and organizational boundaries.

To help foster accessibility, ease of use and wide adoption a Privacy Use Template should have a simple basic structure, while also supporting the in-depth analysis needed to address the complexity of privacy requirements in a software development project. As noted in Section 1, the OASIS Privacy Management Reference Model and Methodology Technical Specification v1.0 (PMRM) supports this Privacy Use Template. It represents a comprehensive methodology for developing privacy requirements for use cases. It enables the integration of privacy policy mandates and control requirements with the technical services and the underlying functionality necessary to deliver privacy and to ensure effective privacy risk management. The PMRM is therefore valuable as the foundation for a comprehensive, standardized and accessible Privacy Use Template.

A PMRM-based template provides:

- a standards-based format enabling description of a specific Privacy Use Case in which personal data or personally identifiable information is involved in a software development project
- a comprehensive inventory of Privacy Use Case/User Story components and the responsible parties that directly affect privacy management and related software development for the Use Case
- a segmentation of Use Case components, or User Stories, in a manner generally consistent with the comprehensive OASIS PMRM v1.0 Committee Specification, and agile methodologies.
- an understanding of the relationship of the privacy responsibilities of software developers in privacy-embedded Use Case/User Story development vis-à-vis other relevant Use Case stakeholders
- insights into *Privacy by Design* requirements throughout the different stages of the privacy life-cycle
- the capability to expose privacy control requirements and their supporting technical services and functionality within a Use Case/User Story boundary and linkages to external privacy management services
- the potential for assessing in an organization essential PbD predicates for software development (privacy training, privacy management maturity, etc.)
- significant value as a tool to increase opportunities to achieve *Privacy by Design* in applications by extracting and making visible required privacy properties.

The template does not specify an implementer's SDLC methodology, development practices or in-house data collection, data analysis or modeling tools.

The Privacy Use Template Components:

1. **Use Case Title**

2. **Use Case Category**

3. **Use Case Description**

4. **Applications associated with Use Case**
   (Relevant applications and products requiring software development where personal data is communicated, created, processed, stored or deleted)

5. **Data subjects associated with Use Case**
   (Includes any data subjects associated with any of the applications in the use case)

6. **PI and PII and the legal, regulatory and /or business policies governing PI and PII in the Use Case**
   - *(The PI and PII collected, created, communicated, processed, stored or deleted within privacy domains or systems, applications or products)*

   - (*The policies and regulatory requirements governing privacy conformance within use case domains or systems and links to their sources)*

7. **Domains, Domain Owners, and Roles associated with the Use Case –** Definitions:
   - ***Domains*** *- both physical areas (such as a customer location or data center location) and logical areas (such as a wide-area network or cloud computing environment) that are subject to the control of a particular domain owner*
   - ***Domain Owners*** *- the participants responsible for ensuring that privacy controls and functional services are defined or managed in business processes and technical systems within a given domain*
   - ***Roles*** *- the roles and responsibilities assigned to specific participants and systems within a specific privacy domain*

8. **Data Flows and Touch Points Linking Domains or Systems**

- *Touch points - the points of intersection of data flows with privacy domains or systems within privacy domains*
- *Data flows – data exchanges carrying PI and privacy policies among domains in the use case*

9. **Systems supporting the Use Case applications**
   (System - a collection of components organized to accomplish a specific function or set of functions having a relationship to operational privacy management)

10. **Privacy controls required for developer implementation**
    (Control - a process designed to provide reasonable assurance regarding the achievement of stated objectives [*Note: to be developed against specific domain, system, or applications as required by internal governance policies, business requirements and regulations]*

11. **Services and 12. Underlying Functionality Necessary to Support Privacy Controls**
    - *Service - a collection of related functions and mechanisms that operate for a specified purpose*

The following illustration (Fig. 4.0) highlights these twelve template components. Note that the template is not a hierarchical model. It recognizes, as does the PMRM on which it is based, the overlapping roles of stakeholders having PbD responsibilities and roles in software development.

Fig. 4.0 Color-coded Grouping of Steps of the Privacy Use Template

Organizational Stakeholders Responsible for Template Development

Responsibilities for contributing to the development of a Use Case and providing information related to specific template components will vary (particularly within a large organization) as illustrated in the following example:

Stakeholder /Domain Owners	Data Subjects	Applications	PI/ PII	Legal/ Regulatory Policies	Domains	Data Flows/ Touch points	Privacy Controls	Services and Technical Functionality
CPO	x		x	x	x		x	
IT Architect					x	x	x	x
Business Analyst	x	x			x			
Team Privacy Champion	x		x	x		x	x	
Senior Developer		x					x	x
Line of Business Owner					x			
Legal Department				x	x			x
CIO								
…								

4.2 Documenting Visual Models for Privacy Requirements Analysis & Design

The current state-of-the-art in industry to document privacy considerations involves one of or a combination of spreadsheet, DFD, and/or UML representation. The output of the *Privacy by Design* Use Template in section 4.1 may be represented in similar ways as well.

4.2.1 Spreadsheet Modeling

Spreadsheets may be used as part of the required specification documentation. Examples of recorded attributes for a software project include:

Description of Personal Data/Data Cluster

Personal Info Category

PII Classification

Source

Collected by

Collection Method

Type of Format

Used By

Purpose of Collection

Transfer to De-Identification

Security Control during Data Transfer

Data Repository Format

Storage or data retention site

Disclosed to

Retention Policy

Deletion Policy

These fields also document several outputs of the *Privacy by Design* Use Case Template. In addition to these fields, spreadsheet tabs can contain DFDs and models as described in the following subsections. Other spreadsheet tabs may contain designer consideration of privacy properties. When there are multiple good ways to design a solution, the option that provides the least exposure of identity, link-ability to other data that can re-identify a user/data subject, or observ-ability of private data or identity, and least complexity is preferred.

4.2.2 Modeling Languages

The section below uses UML for illustration and determination of equivalence in expressive power, in the case that visual software engineers chose not to document with spreadsheets, or to complement their spreadsheet use. An advantage of modeling languages is their expressive power. They enable people, who understand the problem, and people, who design and implement the information technology solutions, to communicate detailed understanding of functional requirements and to clearly represent interactions with multiple stakeholders. Diagramming speeds up the requirements gathering and specification phase in software engineering. In addition, capturing diagrams in formal documentation provide a useful audit trail.

Different modeling languages are used across industries. The software industry uses several with the Unified Modeling Language (UML) being a popular standard. This specification remains agnostic to the software engineers' choice of modeling language.

4.2.2.1 *Privacy by Design* and Use Case Diagrams

This section illustrates how software engineers may *visualize, document, and communicate privacy requirements, select controls, and represent them* in black-box services using high-level *use case diagrams as one of their documentation mediums.*

UML Use Case diagrams are recommended for software engineers to use to easily visualize and document the embedding of privacy into their designs as they abstract out and group details. Use cases are composed of many smaller use case scenarios and/or user stories. As many systems are too complex to be represented in one page, software engineers utilize larger component use cases to hide the system complexity and to handle scale in the use case diagram. There is a similar scaling problem when representing privacy requirements in use case/user story diagrams.

UML is extensible. It may use <<stereotypes>> to reduce the clutter and support scaling as the number of privacy service operations that are required increases. The software engineer may

use a Privacy ServicesContainer or a Container stereotype, for example, as is shown in Fig. 4.1 [Jutla et al, 2013]. The ServicesContainer will host all the privacy services required for a use case diagram and reduce the diagram's complexity by avoiding the clutter of multiple instances of a privacy service on a use case diagram.

By using components that are well understood by modelers, and directly hooked to privacy services, the software engineer can document privacy requirements and selection of privacy controls for the use case scenario, or user stories. An example is provided below.

Example to illustrate the concept of PbD-extended Use Case Diagrams:

In Fig. 4.1, the "super" privacy ServicesContainer is on the communication line between the scientist and the use case scenario. It contains three privacy services: (i) The first privacy service implements the control that requires showing the privacy notice to the scientist, on the use of output data by the program, and obtaining an agreement from her/him. (ii) The second service implements the pseudonymization control for the data before it is used by an application/app. (iii) The third service specifies that anonymization (or de-identification) control is to be applied. As the connection line in Fig. 4.1 from the data scientist is connected to the privacy container, and since the communication line from the container is connected to the sub use case, all privacy services and the controls they represent within the container apply.

Fig. 4.2 shows a more complex UML use case diagram. It is the same scenario as in the previous case, with the addition of a doctor who needs to review recommended treatments, and two further actors. The doctor also needs to be presented with the privacy notice, and the system also needs the doctor's agreement to the conditions specified in the notice that may involve conditions from a patient's consent directive. Data shown to the doctor needs to be pseudonymized. For this case scenario an additional requirement is that the system must communicate with the scientist and also the doctor over secure channels. How these privacy requirements are represented using privacy services implementing controls is shown in Fig. 4.2. As in the previous case the data scientist is connected to the container, signifying that all privacy services apply.

All privacy requirements, specified by privacy services within the container, apply to the scientist and the View alternative treatments use case scenario: communication must be over a secure channel, pseudonymization on input data must apply, privacy notice must be given and agreement obtained, and anonymization must be applied on output data. For the doctor, only three controls apply: communication over the secure channel, pseudonymization of data, and privacy notice and agreement. Anonymization is not applied. Consequently, the doctor actor is connected directly to the applicable privacy controls within the container. Furthermore, the doctor's communication lines need to be labeled to properly identify the connections between the doctor actor, applicable privacy controls, and the doctor's sub use case.

Further in Fig. 4.2, the data scientist actor, as before is connected to the privacy container signifying that all privacy controls within the container apply to the data scientist's interaction with the View alternative patients treatments use case scenario. However, there is additional

detail in that there are two anonymization methods specified within the Anomymization control. Suppose now that the data scientist has requested and received full and extended access to an anonymized version of the big data set in order to troubleshoot a problem issue. As the scientist is connected to the container, and not directly to the control, the default method, k-anonymity with large k, is specified for her. The public researcher is a new actor that accesses data on which a strong anonymization method, based on the concept of l-diversity, is applied. Another new actor is a head nurse who views a specific treatment record – only secure communication is required. The nurse actor is connected to the Security control and then to the View treatment sub use case. The doctor is now connected to two sub use cases. The doctor is connected to the Review Recommended Treatments sub use case, which requires pseudonymization, notice and agreement, and secure connection. He/she is also connected to the View treatment sub use case – in which case only a secure connection is required.

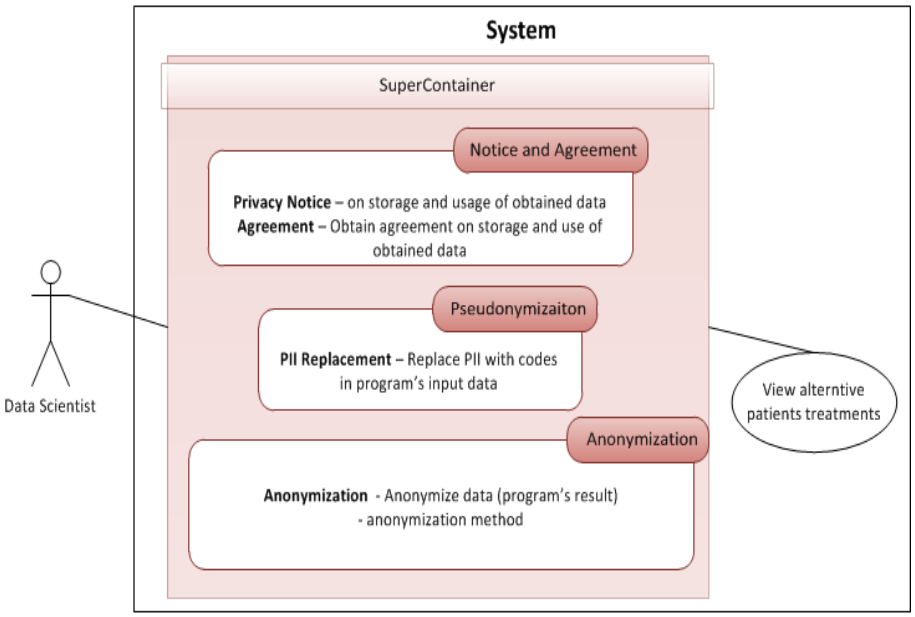

Fig. 4.1 Single-actor use case diagram with privacy services implementing controls [Jutla et al, 2013]

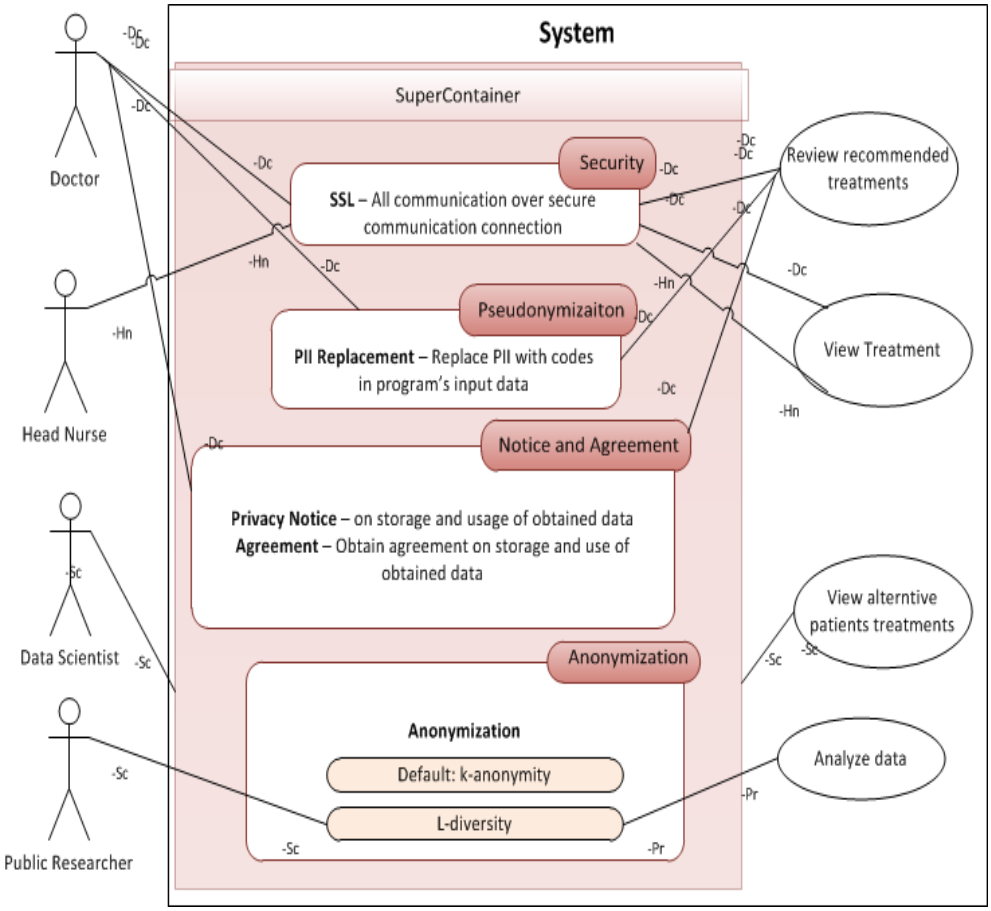

Fig. 4.2 Visualizing Privacy Requirements with multiple-actor use case diagram with privacy services implementing controls [Jutla et al, 2013]

4.2.2.2 *Privacy by Design* and Misuse Case Diagrams

UML Misuse Cases and Misuse Case Diagrams highlight and document the ways actors can violate stakeholder privacy. They add a view from threat modeling (See [Shostack, 2014]).

4.2.2.3 *Privacy by Design* and Activity Diagrams

UML Activity Diagrams document multiple levels of detail. What we call the Business Activity Diagram may be used for the two highest levels of the Zachman Framework [1987]. Software engineers may develop Enterprise Activity Diagrams to show business function relationships at the highest level of the enterprise.

Getting to the next level, they may use Business Activity Diagrams for each project and for each use case. The Activity Diagram shows process relationships and key decisions, and much more information than Use Case Diagrams. Use Case Diagrams are valuable to show the relationships of actors to the various use cases and privacy controls. But the Business Activity Diagram is more valuable for detailed analysis.

Figure 4.6 illustrates the Hospitality company's Business Activity Diagram as a process modeling tool. We add the Activity Diagram Object icon to show major data attributes tied to processes and decisions to highlight and document where privacy concerns need to be addressed. Business activity diagrams consider and express privacy details without needing to use UML extensions. Figure 4.7 illustrates the use of Business Activity Diagrams with privacy impacting data attributes.

System Activity Diagrams support the design and documentation of modules within a system design. The System Activity Diagram (Figure 4.8) is augmented with an UML Activity Diagram Note icon to show privacy principles and to show which modules address which privacy principles (see Fig. 4.9).

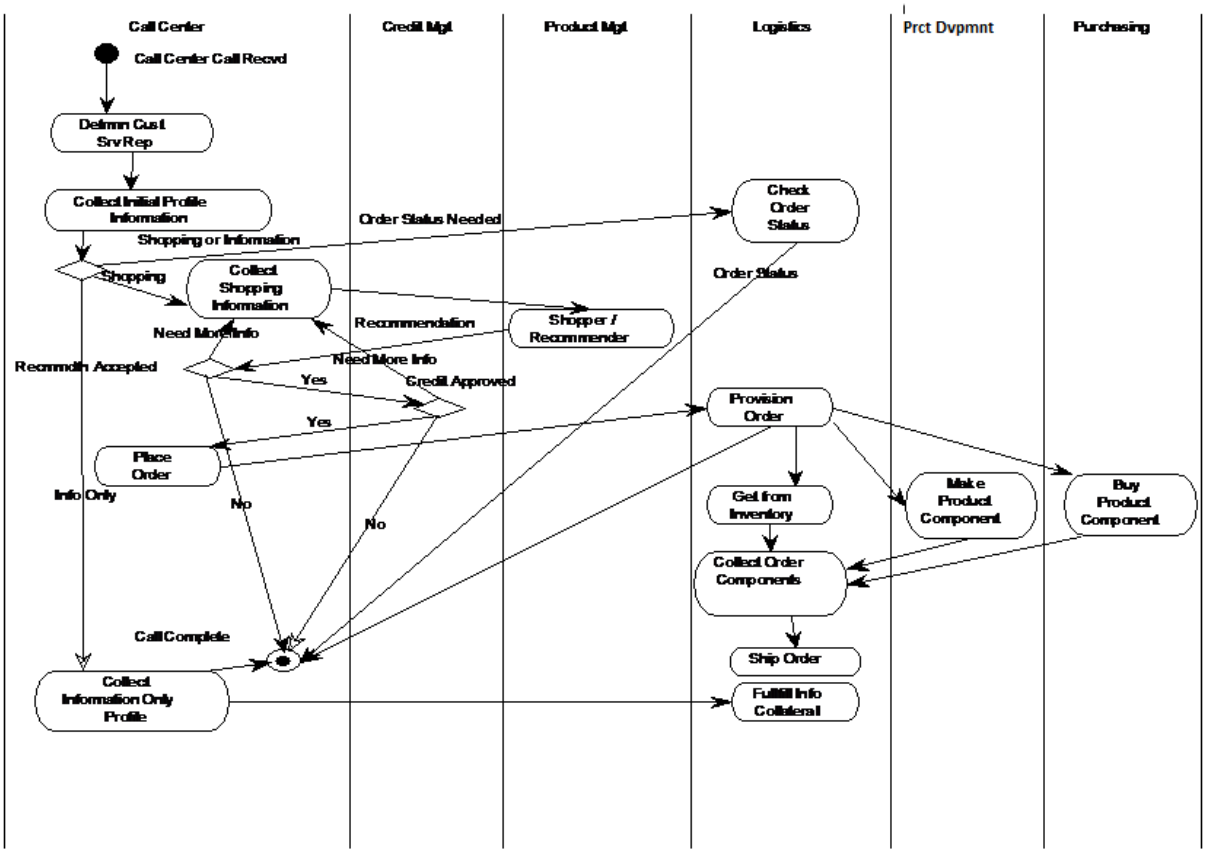

Fig. 4.6. Business Activity Diagram for a Vacation Planning project in a Hospitality Company [Dennedy et al 2014]

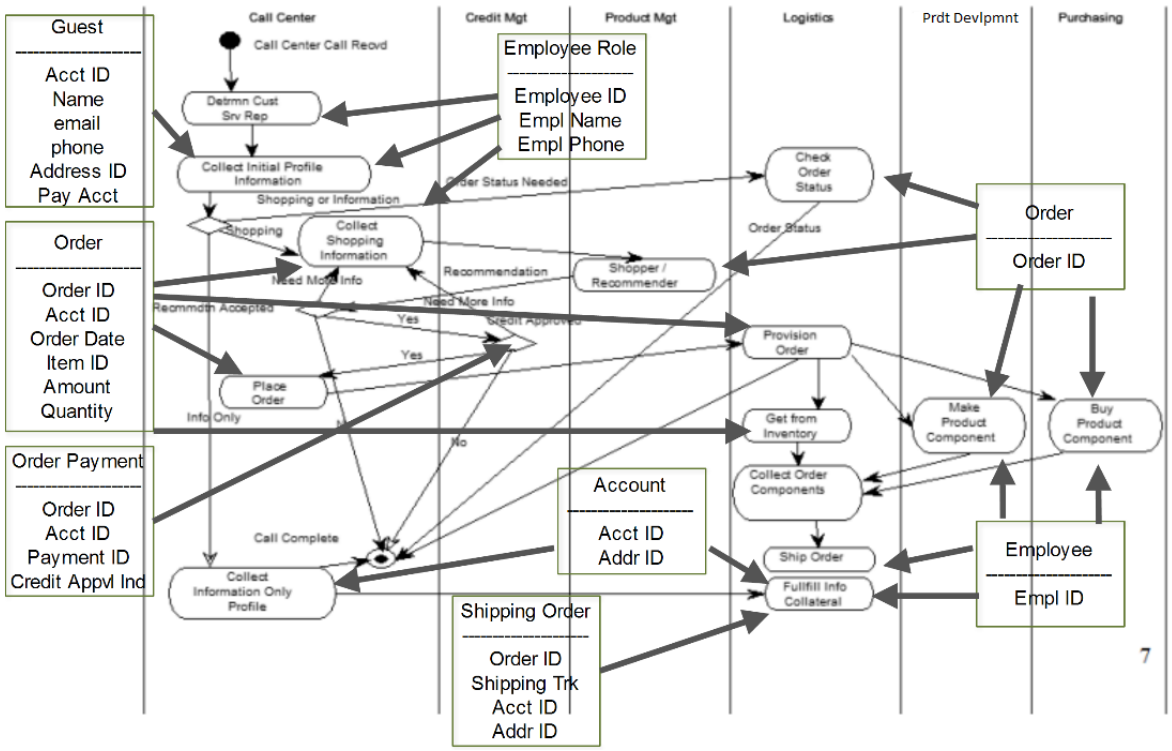

Fig. 4.7. Visualizing Privacy-Impacting Attributes on a Business Activity Diagram [Dennedy et al 2014]

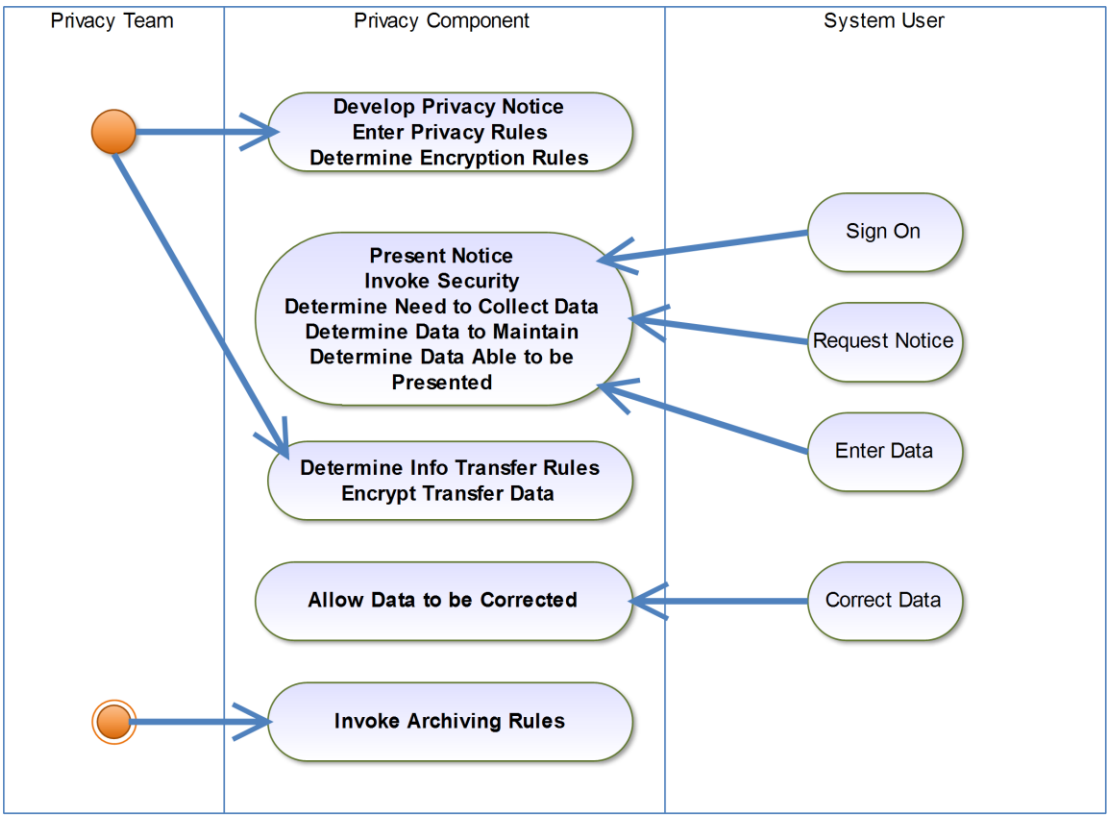

Fig. 4.8 Privacy Components [Dennedy et al 2014]

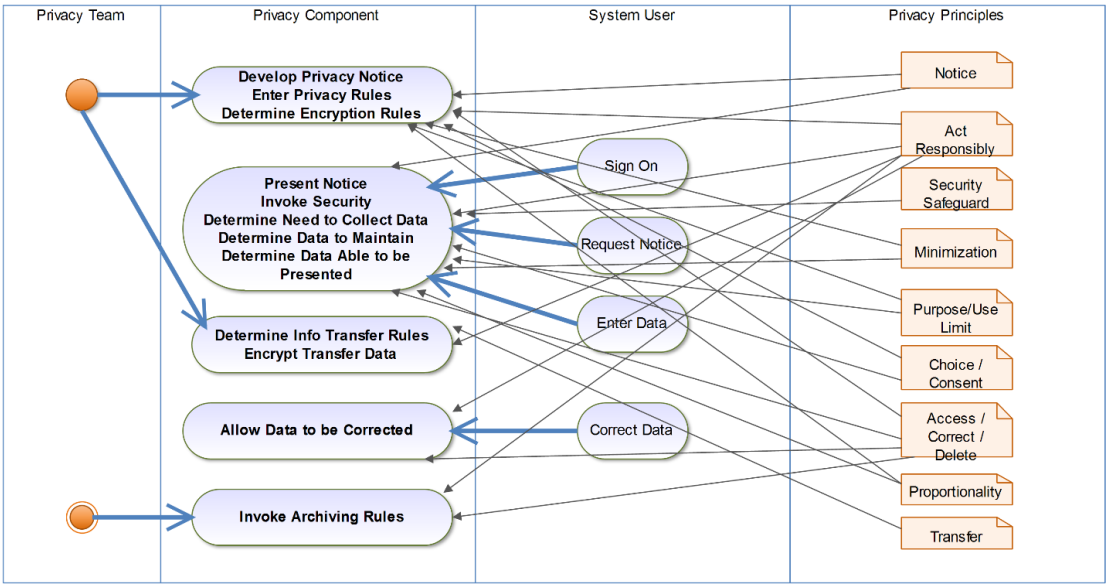

Fig. 4.9 Mapping *Privacy by Design*, GAPP, of FIPPs Principles to Privacy Components [Source: Dennedy et al 2014]

4.2.2.4 *Privacy by Design* and Sequence Diagrams

Software engineers may also visualize and document privacy requirements by embedding privacy services among functional requirements in UML sequence diagrams (see Fig. 4.10).

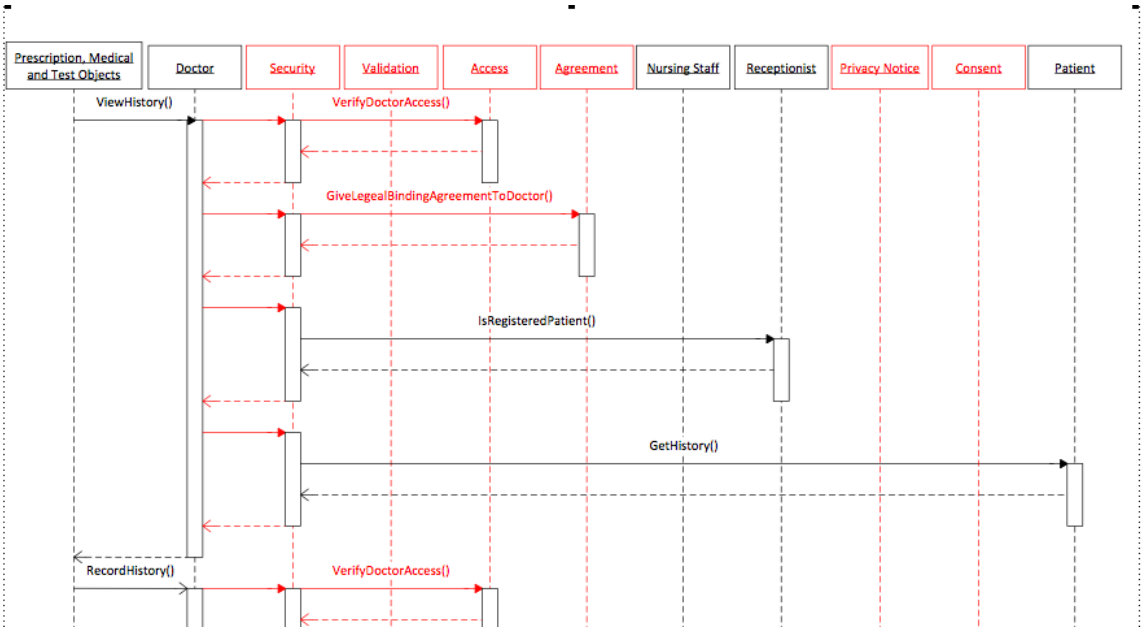

Fig. 4.10 Visualizing Privacy Services in a UML Sequence Diagram

4.3 *Privacy by Design* and Privacy Reference Architecture

A high-level, full stakeholder-view, privacy reference architecture (Fig. 4.11) is provided for customization by a software organization, i.e. any organization that creates and/or uses software to collect and manage client and other stakeholder data. Functional software (e.g. an online social network) that collects data is shown at the bottom level. Functional privacy services (layer 1) are integrated in the functional software through APIs. These privacy services then provide data to a management or integration service layer. These middle-layer services bridge to organizations' business-related privacy services on collected data at composite layer-3.

Organizations may adopt and customize variants of such a privacy reference architecture, *shrinking* or *growing* it depending on their areas of emphases and privacy maturity level. The eight PMRM services of Security, Agreement, Access, Usage, Validation, Interaction, Certification, and Enforcement form a core architectural pattern, as they are repeatable at touch points, i.e. at the interface of two or more stakeholders, applications, systems, data owners, or domains [Jutla et al, 2014]. Additional privacy services, such as for data minimization, complement them at the user interface layer. Data repositories are not shown in Figure 4.11, due to its services-focus, but software engineers will drill down, possibly guided by this SOA reference architecture, or an equivalent non-SOA architecture, and incorporate data repositories in further architectural viewpoints.

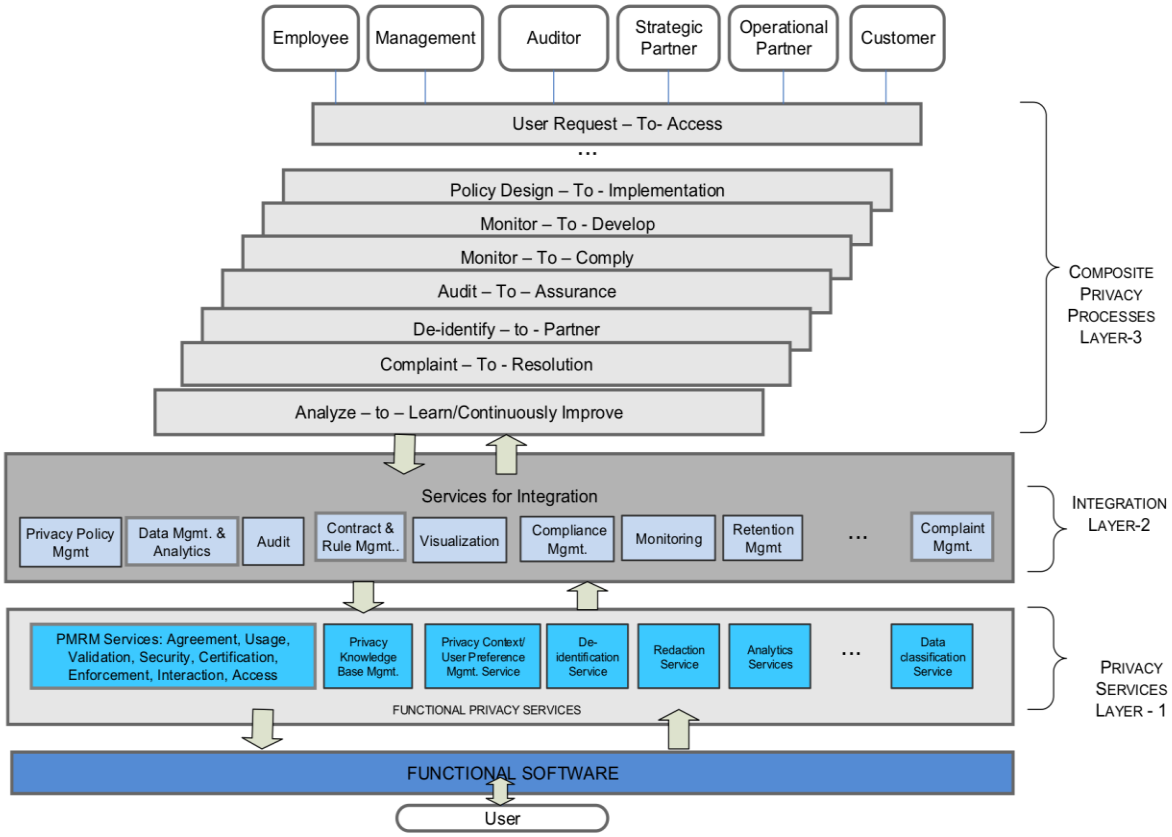

Fig. 4.11. Privacy Reference Architecture [Jutla 2014, 2014a]

4.3.1 Privacy Properties

This specification requires software engineers to consider and create software that satisfies a reasonable and attainable subset of privacy properties. Privacy properties may be divided into user-level and systems-level properties. At the user level, we use the 7Cs (Comprehension, Consciousness, Choice, Consent, Context, Confinement, and Consistency [Jutla and Bodorik, 2005] – see Table 4.1 below) as standardized privacy properties for solutions to meet to show respect for individuals as per the seventh PbD Principle. Note that the joint Canadian Institute of Certified Accountants/American Institute of Chartered Public Accountants effort uses 7 criteria for measuring choice and consent in its privacy maturity model [CICA 2014].

At the systems-level, the use of at least ATOIL – Auditability, Traceability, Observ-ability, Identifi-ability, and Link-ability properties [Jutla et al, 2014, Jutla 2014b] is required in architecting privacy in software. The Best Information through Regional Outcomes [BIRO 2009] Health project defines the identify-ability privacy property "as a measure of the degree to which information is personally identifiable"; the link-ability privacy property as "a measure of the degree to which the true name of the data subject is linkable to the collected data element"; and observ-ability privacy property as the "measure of the degree to which link-ability and

identify-ability are affected by the use of the system" over various contexts as system-level properties. We adopt these definitions. For privacy, the traceability privacy property is important to understand privacy threats or leaks arising from data flowing among software components and systems; the auditability privacy system property is desirable so that the software engineer and her/his company is assured that the system is implementing and executing privacy solutions correctly.

Table 4.1 User-level privacy properties

Comprehension (User understanding of how PII is handled)	Users/data subjects should *understand* how personal identifiable information (PII) is handled, who's collecting it and for what purpose, and who will process the PII and for what purpose across software platforms. Users/Data subjects are entitled to visibility - to know all parties that can access data subjects' PII, how to access/correct data, the limits to processing transparency, why the PII data is being requested, when the data will expire (either from a collection or database), and what happens to it after that. This category also includes legal rights around PII, and the implications of a contract when one is formed.
Consciousness (User awareness of what is happening and when)	Users/data subjects should be *aware* of when data collection occurs, when a contract is being formed between a user and a data collector, when data subjects' PII is set to expire, who's collecting the data, with whom the data will be shared, how to subsequently access the PII, and the purposes for which the data is being collected.
Choice (To opt-in or out, divulge or refuse to share PII)	Data subjects should have *choices* regarding data collection activities in terms of opting in or out, whether or not to provide data, and how to correct their data.
Consent (Informed, explicit, unambiguous)	Data subjects must first consent (meaning informed, explicit, unambiguous agreement) to data collection, use, and storage proposals for any PII. Privacy consent mechanisms should explicitly incorporate mechanisms of comprehension, consciousness, limitations, and choice.
Context	Users/data subjects should/must be able to *change privacy preferences* according to context. Situational or physical context—such as crowded situations (for example, when at a service desk where several people can

(User adjusting preferences as conditions require)	listen in on your exchange when you provide a phone number, or when you are in the subway with cameras and audio on wearables around you)—is different from when you perform a buy transaction with Amazon.com or provide information to an application registered with an aggregator that sells to advertisers. Data also has context (such as the sensitivity of data, for example, financial and health data) could dictate different actions on the same PII in different contexts.
Confinement (Data minimization, proportionality, and user-controlled re-use of data)	Users/data subjects must/should be able to *set/request limits* on who may access their PII, for what purposes, and where and possibly when/how long it may be stored.
Consistency (User predictability of outcome of transactions)	Users should *anticipate with reasonable certainty* what will occur if any action involving their PII is taken. That is, certain actions should be predictable on user access of PII or giving out of PII.

4.4 *Privacy by Design* and Design Patterns

4.5 Coding / Development

This section describes software engineering tools and techniques for operationalizing *Privacy by Design* into the coding / development phase of the software development life cycle.

[Note that the name “coding / development” is used instead of “implementation” in order to prevent confusion with implementation in the sense of end-user deployment. Tabled for a future version]

4.6 Testing / Validation

This section describes software engineering tools and techniques for operationalizing *Privacy by Design* into the testing / validation phase of the software development life cycle.

Validation and Verification – Develop tests at the same time you develop requirements.

Reference OWASP for Penetration testing/ Code Review including code scanning (security)]

4.6.1 *Privacy by Design* Structured Argumentation

4.7 Deployment Phase Considerations

This section describes privacy issues and methods for operationalizing *Privacy by Design* in the deployment phase of the software development life cycle. It is not intended to produce strict documentation guidance. Rather, it is only meant to offer considerations to be taken into account by software engineers.

4.7.1 Fielding

4.7.2 Maintenance

4.7.3 Retirement

See subsection 3.8.

4.8 Privacy Checklists

In addition to using Table 3.1 as a checklist, software engineers should use the third column entries in Table 2.1 of the PbD-SE specification [PbDSE-01] as a checklist for documentation, which should be generated within organizations producing software, and available to auditors.

Appendix A. Acknowledgements

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Participants:

Peter Brown, Individual
Kim Cameron, Microsoft
Fred Carter, Office of the Information & Privacy Commissioner of Ontario, Canada
Ann Cavoukian, Office of the Information & Privacy Commissioner of Ontario, Canada
Les Chasen, Neustar
Michelle Chibba, Office of the Information & Privacy Commissioner of Ontario, Canada
Frank Dawson, Nokia
Mike Davis, Individual
Eli Erlikhman, SecureKey technologies, Inc.
Ken Gan, Ontario Canada Lottery and Gaming Corporation
Sander Fieten, Individual Member
Jonathan Fox, Intel Corporation
Frederick Hirsch, Nokia
Gershon Janssen, Individual Member
Dawn Jutla, Saint Mary's University
Antonio Kung, Trialog
Kevin MacDonald, Individual
Drummond Reed, XDI.org
John Sabo, Individual
Stuart Shapiro, MITRE Corporation
Aaron Temin, MITRE Corporation
Colin Wallis, New Zealand Government
David Weinkauf, Office of the Information & Privacy Commissioner of Ontario, Canada
Michael Willett, Individual

Appendix B. Revision History

Revision	Date	Editor	Changes Made
00	10 July 2013	Ann Cavoukian Fred Carter	Initial Working Draft Outline
00	20 Aug 2013	Ann Cavoukian Fred Carter	Revisions to document structure, content added to sections 1 and 2
00	07 Oct 2013	Ann Cavoukian Fred Carter	Incorporate TC member comments and suggested revisions to v02; modify document structure, add new PbD content. Revisions accepted by Committee 30 October 2013 as basis for next revision of working draft.
00	06 Mar 2014	Dawn Jutla	Introductory paragraphs added. Revised Section 3 as a methodology to produce *Privacy by Design* documentation for software engineers; steps refined and re-ordered Created and added new Section 4 (now section 2). Created and added new mapping of PbD principles to sub-principles, privacy services, and documentation. (Section 4 is removed from this Annex and placed in Specification's new Section 2) Restructuring of Section 5. Insertion of new materials
00	07/08 Mar 2014	Dawn Jutla	Refinement of Sections 3 and 4 with TC at March 7 and 8's face to face meetings.
00	10 Mar 2014	Dawn Jutla	Addition of further steps in methodology
00	07/08 Mar 2014 15 June 2014	Sander Fieten	Suggest conformance verbs for Table 4.1. (removed from this Annex – put in Specification as Table 2.1) Edits and comments in removed section 4.

00	7 Jan 2014 8 Mar 2014	Frank Dawson	Provided Spreadsheet modeling for Section 5 (now section 4). Contributions to methodology in Section 3.
00	21 Mar 2014	Dawn Jutla	Filled in initial text for all 10 methodological steps, and created sample RACI table (Fig. 3.1) for the methodology in Section 3.
	19 Apr 2014		Substantially revised and completed mapping of PbD-principles to documentation in Section 4 Table 4.1 (now Section 2, Table 2.1). Removed Privacy services mapping from this version. Added Visual Modeling sections 4.2, 4.2.2, 4.2.2.1, 4.2.2.2, 4.2.2.4 and Architecture section 4.3 and 4.3.1.
00	24 April 2014	Ann Cavoukian Fred Carter	Completion of Section 2 with 7 PbD principles and sub-principles. Substituted new descriptions of sub-principles into Table 4.1 (now removed to PbD-SE Specification as Table 2.1)
00	29 April 2014	John Sabo	Added Privacy Use Template in Section 4.1.
00	9 May 2014	Jonathan Fox	Added bulleted points in Sections 3.1 and 3.5.
00	9 May 2014	Tom Finneran	Created and added Section 4.2.2.3
00	10 May 2014	Dawn Jutla	Substantial edits throughout sections 3, old section 4, & 5 (now new section 4).
00	31 May 2014	Dawn Jutla	Filled in section 1.4, and multiple edits throughout all former sections 1, 3, 4, 5, & 6 of document.
00	3 June 2014	John Sabo	Feedback and edits throughout
00	10 Jun 2014	Dawn Jutla	Added Fred's edits to John's.
00	17 June	Dawn Jutla	Combined TC edits (Jonathan, Stuart, Sander, John, Fred, Colin) and minor edits for sections 1,

			3, 4, and 5
00	15 June 2014	Fred Carter	Minor revisions to sections 2 and former section 4 + new conformance section 6 (now moved to spec)
	17 June 2014	Fred Carter	Edits to Section 2 and minor edits to sub-principles in former table 4.1.
00	17 Jun 2014	Kim Cameron	Major reorganization of specification – split into specification and Annex
01	18 Jun **THIS VERSION AS ANNEX**	Dawn Jutla	Processed TC input for re-organization of document into specification and Annex How-to Guide to address specification adoption. MAJOR CONTENT PRODUCED FROM ALL ABOVE REVISIONS PRESENT IN THIS ANNEX.
01	22 Jun 2014	Dawn Jutla	Processed all TC edits in ANNEX

