Privacy technologies need to go to the gym on the challenges of privacy engineering in an AGILE world

Carmela Troncoso

Joint work with Seda Güerses (TU Delft) and Blagovesta Pirelli (EPFL)
Why we started working on privacy engineering

Privacy Embedded into Design

“Privacy by design is embedded into the design and architecture of IT systems [...]. It is not bolted as an addon, after the fact. The result is that privacy becomes an essential component of the core functionality being delivered. Privacy is integral to the system without diminishing functionality".
Why we started working on privacy engineering

Actually... “Data Protection by design and by default”

“the controller shall [...] implement appropriate technical and organisational measures [...] which are designed to implement data-protection principles [...] in order to meet the requirements of this Regulation and protect the rights of data subjects.”
Why we started working on privacy engineering

Privacy by Design principles

1. Proactive not Reactive; Preventative not Remedial
2. Privacy as the Default Setting
3. Privacy Embedded into Design
4. Full Functionality: Positive-Sum, not Zero-Sum
5. End-to-End Security — Full Lifecycle Protection
6. Visibility and Transparency — Keep it Open
7. Respect for User Privacy — Keep it User-Centric

Companies should promote consumer privacy throughout their organizations and at every stage of the development of their products and services. Companies should incorporate substantive privacy protections into their practices, such as data security, reasonable collection limits, sound retention practices, and data accuracy. Companies should maintain comprehensive data management procedures throughout the life cycle of their products and services.
Why we started working on privacy engineering
How can we make the principles in Academia accessible to Engineers (and Educators)?

They “don’t” need it
Two case studies

Anonymous e-petitions: no identity attached to petitions

Privacy-preserving road tolling: no fine grained data sent to server

The KEY is “data minimization”

Related to a key regulation principle, well in synch with policy makers!
The key is "data minimization," but it's not "data" that is minimized (in the system as a whole). Data is kept in user devices, sent encrypted to a server (only client has the key), distributed over multiple servers.

The KEY is “data minimization”

but, it’s not “data” that is minimized (in the system as a whole)

data is kept in user devices
sent encrypted to a server (only client has the key)
distributed over multiple servers ...

“data minimization”

is a BAD metaphor
Engineering Privacy by Design 2.0
Unpacking Data Minimization

Minimize

TRUST ASSUMPTIONS placed on other entities => PRIVACY RISKS

Minimize Collection
Minimize Disclosure
Minimize Linkability
Minimize Centralization
Minimize Replication
Minimize Retention

Risk is understood by businesses &
Support proportionality for policy makers!

The turn to agile

Privacy After the Agile Turn

Seda Gürses and Joris van Hoboken

In this chapter, Seda Gürses and Joris van Hoboken explore how recent paradigmatic transformations in the production of everyday digital systems are changing the conditions for privacy governance. Both in popular media and in scholarly work, great attention is paid to the privacy concerns that surface once digital technologies reach consumers. As a result, the strategies proposed to mitigate these concerns, be it through technical, social, regulatory or economic interventions, are concentrated at the interface of technology consumption. The authors propose to look beyond technology consumption, inviting readers to explore the ways in which consumer software is produced today. By better understanding recent shifts in software production, they argue that it is possible to get a better grasp of how and why software has come to be so data intensive and algorithmically driven, raising a plethora of privacy concerns. Specifically, they highlight three shifts: waterfall to agile development methodologies; shrink-wrap software to services; and, from software running on personal computers to functionality being carried out in cloud. They shorthand the culmination of these shifts the “agile turn”. With the agile turn, the complexity, distribution and infrastructure of software has changed. What are originally intended to be techniques to improve the production of software development, e.g., modularity, agility, come to also reconfigure the way businesses in the sector are organized. In fact, the agile turn is so tectonic, it
Engineering Privacy by Design 2.0

Unpacking Data Minimization

Minimize Collection
Minimize Disclosure
Minimize Centralization
Minimize Linkability
Minimize Replication
Minimize Retention

TRUST

ASSUMPTIONS placed on other entities => PRIVACY RISKS

How to use them
The turn to agile

waterfall model → agile programming

shrink wrap → services

PC → cloud
Why does this change anything? Isn’t it still about minimizing **TRUST**?
Why does this change anything? Isn’t it still about minimizing TRUST?

Waterfall Software Development

- Requirements
- Design
- Development
- Verification
- Deployment
- Maintenance

Management agile cycle:
- customer
- management
- marketing
- ...

Development agile cycle:
- add functionality
- integrate
- test
Why does this change anything? Isn’t it still about minimizing Trust?

Where are the specs?!?

And trust on...? How do we reason?
But c’mon we are computer scientists, can’t we just fix it?

PETs are not designed to evolve!

PETs are hard to compose!

PETs require full control on design!

PETs are not Agile....
Why did this happen?

Privacy technology design and privacy engineering have implicit conceptions of software engineering practice that do not match current reality of the practice in the wild.

**assumption:** system designer has complete control of each and every component
- designer may have to integrate third party services
- microservice may be applied in multiple contexts

**assumption:** the system is monolithic
- integration and composition is difficult but necessary

**assumption:** the system is static
- find ways to check privacy properties hold under change
Privacy technology design and privacy engineering have implicit conceptions of software engineering practice that do not match current reality of the practice in the wild.

**Assumption:** System designer has complete control of each and every component.
- Designer may have to integrate third party services.
- Microservice may be applied in multiple contexts.

**Assumption:** The system is monolithic.
- Integration and composition is difficult but necessary.

**Assumption:** The system is static.
- Find ways to check privacy properties hold under change.
Systematic study of academic proposals

4 last years (2015-2018) of 3 top conferences (S&P, SEC, NDSS)
87 papers related to privacy (manual selection)

Two aspects

What do we design?

Systems, Components, Protocols, Evaluation tools, Policy analysis

Are designs aware of engineering needs?

Systematize, Generalize, Framework, Best Practices, Context, Dynamism
What do we design?

- Systems: 9
- Component: 27
- Protocol: 10
- Evaluation tool: 43
- Policy analysis: 5
Are designs aware of engineering needs?

- Systematizes: 11
- Generalizes: 19 (21%)
- Framework analysis/design: 11
- Sets best practices: 18
- Considers context: 11 (12%)
- Considers dynamism: 8 (9%)
Diving deeper

27 out of 87 papers do not mention any engineering factor

37 out of 87 have software artifacts (we did not look at ease-to-use)
14 of these do not consider engineering factors

Out of 27 components only 9 consider context and 3 dynamism
What about engineering-support? Are they aware of PETs and Agile?

Systematic study of engineering methodologies

4 standards (ISO, NIST, OASIS, PRMR)

- ISO TR 27550 - Privacy engineering for system life cycle processes
- Privacy by Design Documentation for Software Engineers Version 1.0/OASIS
- NIST An Introduction to Privacy Engineering and Risk Management in Federal Systems
- Privacy Management Reference Model and Methodology (PMRM) Version 1.0

8 academic proposals (Software Engineering and Security & Privacy)

- A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements
- Engineering privacy
- Security and privacy requirements analysis within a social setting
- PRIAM: a privacy risk analysis methodology
- Protection goals for privacy engineering
- Privacy design strategies
- Engineering Privacy by Design/Engineering Privacy by Design Reloaded
- Applying Privacy by Design in Software Engineering - An European Perspective
What about engineering-support? Are they aware of PETs and Agile?

**Vision of the system**: monolith or service-oriented

*Standards*

3 out of 4 consider services, but also see the system as a monolith
The fourth does not consider a monolith, but ignores services

*Academia*

4 out of 8 consider services, but ALL see the system as a monolith
(Hoepman’s strategies are a bit more flexible)

**Recommendations nature**: heuristics vs. checklists

*Standards*

Checklist and recipe

*Academia*

2 out of 8 heuristics, rest checklist-y
What about engineering-support? Are they aware of PETs and Agile?

**Evolution**: considers integration and/or dynamicity

*Standards*
- ISO considers integration, OASIS considers dynamism (2 out 4)

*Academia*
- None.

**Privacy approach**: risk vs. goal oriented, threat modeling

*Standards*
- ISO considers all, the rest are risk oriented without threat modeling

*Academia*
- Implicit approaches and threat modeling (3 out of 8)
- Half and half on the rest
What about engineering-support? Are they aware of PETs and Agile?

**PET Awareness**: Maps to PETS or Data minimization

*Standards*
- ISO mentions both, the rest does not

*Academia*
- 3 explicitly map to PETS and only 4 talk about minimization
What about engineering-support?
Are they aware of PETs and Agile?

**Context aware**: considers deployment environment

*Standards*
- All of them talk about it (organization-oriented)

*Academia*
- Half of them (also organization)

**Who it speaks to**: organization, engineers, researcher

*Standards*
- Mostly organization, some engineering-oriented comments

*Academia*
- Mixed bag
What do we do now?

Agile PETs!

Train the trainer!
How does a good gym for PETs look like? aka A Wishful Research Agenda

**Updatable PETs**
Can we design technology that can be easily changed?

**New encodings for PETs inputs**
If we can’t change PETs... can we reuse them with different inputs to provide more functionality?

**Composable PETs (and privacy definitions)**
If we can’t change PETs... can we compose them to provide more functionality? Can we have easier security/privacy composition? Is modularity possible?

**Agile evaluation frameworks**
Let us assume PETs can change, evaluation tools need to follow! Can we make (unit) tests that evolve and can be integrated in the development cycles?

**Revamping PETs**
If we can’t change PETs... can we make the cycle lighter?
TL;DL

Agile service-oriented development changes the rules of the Privacy game

Software Engineering Practices MUST be part and parcel of the Privacy (Engineering) Research

PETs designers need to look beyond the design to where the design needs to be integrated

There are many exciting research lines opening up!
Thanks!

Software Engineering Practices MUST be part and parcel of the Privacy (Engineering) Research

Paper to come soon!!

carmela.troncoso@epfl.ch
@carmelatroncoso
www.carmelatroncoso.com