University of Cyprus

Integrating GDPR purpose compliance into software system design

Anna Philippou

Joint work with: E. Vanezi, D. Kouzapas, and G. Kapitsaki

Department of Computer Science, University of Cyprus

March 19, 2026

Presentation at Università degli Studi di Urbino Carlo Bo, 19 March 2026

Talk Synopsis

1. Introduction
2. Purpose Aware Multi Party Session Types
    i. MPSTs – A Quick Introduction
    ii. The Type Language
    iii. The Modeling Language
    iv. Typing Purpose Compliance
3. Integrating into the SE Cycle
4. Conclusions and Future Work

1. Introduction

The notion of privacy

- Technology gives rise to new concerns
- New practices relating to the handling of personal information
  - Databases, social networks, banking, etc.
- Challenges
  - Provide solid foundations for a rigorous understanding of privacy rights, threats, and violations
  - Propose frameworks and techniques for developing systems that conform to their privacy requirements

The GDPR

- EU General Data Protection Regulation (GDPR):
  - multiple principles and rights regarding personal data collection, storage, and processing
  - Personal data is the property of the individual
- Existing and newly-developed systems handling data of EU citizens
  - Organizations must adopt, test and maintain, and be prepared to demonstrate such compliance to regulators
- Non-compliance handled with hefty fines and sanctions

When does a system comply to the GDPR?

- Data Minimization & Purpose Limitation
  - Only necessary data and only for specific uses/purposes
- Transparency & Accountability
  - Provide clear notices about what/who/why uses/collects data (include logs)
- Lawful Basis for Processing
  - Consent/contract
- Support data subject rights
- No Solely Automated Decisions

⇒ Together these principles aim to ensure that data processing is lawful, fair, transparent, and respectful of individual privacy rights

Privacy by Design

- Privacy should be incorporated into a system by default and should be a priority from the beginning of a system's design [Cavoukian, 2008; Data Protection and Privacy Commissioners, 2010]
- Data Protection by Design and by Default: "implementing the appropriate technical and organizational measures in order to meet the requirements of the Regulation and protect the rights of data subjects" [Article 25(1), GDPR]
- Proactive consideration of privacy issues
  - ensuring a system's compliance should take place from the design phase and possible pitfalls should be anticipated and prevented before they materialise, rather than remedied on a reactive basis

The notion of purpose

- Purpose
  - GDPR: Article 5 “Principles relating to processing of personal data”
    - 5(1b) Purpose Limitation: “Personal data shall be collected for specified, explicit, and legitimate purposes and not further processed in a manner that is incompatible with those purposes”
  - U.S. Privacy Act
  - Canada’s Federal Privacy Act
  - U.S. Health Insurance Portability and Accountability Act (HIPAA)

- Example
  - A doctor may process a patient’s medical data for treatment, while the accounting department may use the patient’s address for issuing an invoice

The notion of purpose

- Unlike GDPR rights such as access or erasure, the purpose limitation principle constitutes a behavioural constraint on how personal data may be processed throughout the entire system execution
  - Purpose limitation cannot be captured as a simple rule or permission check but requires dedicated modelling and validation support

- No formal definition of purpose
  - Most existing models treat purposes using textual descriptions bearing little or no semantics

- Need for semantics
  - Allow users to understand how their data is being processed
  - Enable the analysis that a system complies to purpose-related requirements

Objectives

- Propose foundations and tools for reasoning about purpose compliance
- Develop formal methodologies that can be employed in the software engineering cycle for developing privacy-aware models and systems
  - Data Protection by design
  - Provably compliant systems

Our proposal

- Purpose semantics using an action-based approach
  - Markov Decision Processes, Petri nets, business processes – auditing, model checking [Tschantz,Datta,Wing, 2011, Jafari,Safavi-Naini,Fong, Barker, 2014, Riahi Khosravi,Ghassemi, 2017, Basin, Debois, Hildebrandt 2018]
- A processing purpose
  - can be expressed as a workflow of actions and interactions between system entities, involving personal data exchange, storing, reading, and writing
  - purpose restricts not only which data may be processed, but also when, by whom, and under which interaction patterns
- Example
  - Purpose: Heart rate monitoring
  - Private data: Heart-rate data
  - Workflow: Heart-rate data are collected by a wearable and regularly transmitted to the hospital. The monitoring system analyses the data: if normal, it is kept for routine review; if abnormal, an alert is sent to a clinician
- Proposal: Purpose-aware multiparty session types

2. Purpose Aware Multi Party Session Types

i. MPSTs – A Quick Introduction

Session Types – Motivation

- Modern distributed software consists of communicating processes:
  - Microservices
  - Actors
  - distributed workflows
  - network protocols
- These systems follow structured communication protocols.

```
Client → Server : request
Server → Client : response
```

- Traditional type systems check:
  - data types
  - function signatures

  but not communication protocols

  ! Result: communication mismatches, deadlocks, unexpected messages

Session Types

- Session types are types for communication protocols
- They act as a contract, ensuring each process follows the expected communication order
- They describe the sequence of messages exchanged during a session

```
Client: !request . ?response . end
Server: ?request . !response . end
```

- Meaning:
  - ! : send
  - ?: receive
- Guarantees:
  - communication safety
  - protocol fidelity

From binary to multiparty

- Originally introduced for two participants (binary session types) [Honda 1993]
- Subsequently extended to Multiparty Session Types (MPST), which extend session types to involve more than two participants (roles) [Honda, Yoshida, Carbone, 2008]
- Example workflow:

```
Buyer → Seller : order
Seller → Bank : payment_request
Bank → Seller : confirmation
Seller → Buyer : receipt
```

Global Types

- A global type describes the entire conversation among the various roles:

```
G = Buyer → Seller : order.
    Seller → Bank : payment_request.
    Bank → Seller : confirmation.
    Seller → Buyer : receipt.
    end
```

- It can be projected into a local type for each participant

Projection to Local Types

- Projection produces the behaviour expected for each role
- Example:

Buyer

```
!Seller : order.
?Seller : receipt.
end
```

Bank

```
?Seller : payment_request.
!Seller : confirmation.
end
```

Seller

```
?Buyer : order.
!Bank : payment_request.
?Bank : confirmation.
!Seller : receipt.
end
```

Implementing a well-typed protocol

- MPSTs define multiparty conversations from a global perspective and provide means for checking protocol compliance
- They can be used to verify that a model/implementation satisfies the expressed protocol
- Models can be expressed in
  - Process calculi (e.g., π-calculus with session types)
  - Behavioral models (actor workflows, choreography languages)
  - Domain-specific notations (e.g., Scribble)
- Checking takes place staticaly through type checking that ensures:
  - Each participant follows its protocol role
  - Communications conform to the global session type

Modeling and Type checking

Putting it all together:

- Session types → specify protocols
- Modeling languages → express (candidate) implementations
- Type checkers → verify fidelity between model and protocol
- The result:
  - Communication correctness
  - Safe composition of distributed components
  - Formal foundation for code generation & testing

2. Purpose Aware Multi Party Session Types

ii. The Type Language

MPSTs for modeling purpose

- We propose MPSTs to capture a purpose as the interation protocol among a set of participantsto achieve the intended objective
- Private data is integrated into the framework as a basic building block
- Private data notation: $i \otimes c$
  - i is the data subject, setting $i = \_$ implies that the data is anonymized
  - c is the data

  Examples: Alice⊗address, Bob⊗credit_card
- Private data is stored in data stores, e.g. cells of a database
  - Notation $r[i \otimes c]$
  - The manipulation of private data takes place through references to their store

The types syntax

- In the language we assume:
  - A set of participating roles: p, q, …
  - A set of base types, g, such as int, bool, etc
  - A type t_i pertaining to the identities of the data subjects
  - A set of annotations, α, β, γ, … allowing to reason about private data access

Purpose-aware Session Types

$$
\begin{array}{lcll}
 & & \alpha, \beta, \gamma, \ldots & (\textit{annotations}) \\
\mathsf{g} & ::= & \texttt{int} \mid \texttt{bool} \mid \ldots & (\textit{Ground Types}) \\
\mathsf{t_i} & ::= & \mathsf{i} \mid \_ & (\textit{Id Types}) \\
\mathsf{U} & ::= & \mathsf{g} \mid [\mathsf{t_i} \otimes \mathsf{g}]^{\alpha} & (\textit{Exchange Types}) \\
\mathsf{G} & ::= & \mathsf{p} \to \mathsf{q}{:}\langle \mathsf{U} \rangle.\, \mathsf{G} & (\textit{Value Exchange}) \\
 & \mid & \alpha \to \mathsf{p}{:}\langle \mathsf{t_i} \otimes \mathsf{g} \rangle.\, \mathsf{G} & (\textit{Personal Data Read}) \\
 & \mid & \mathsf{p} \to \alpha{:}\langle \mathsf{t_i} \otimes \mathsf{g} \rangle.\, \mathsf{G} & (\textit{Personal Data Storage}) \\
 & \mid & \mathsf{p} \to \mathsf{q}{:}\langle \{\ell_i : \mathsf{G}_i\}_{i \in I} \rangle & (\textit{Select/Branch}) \\
 & \mid & t \mid \mu t.\mathsf{G} & (\textit{Recursion}) \\
 & \mid & \mathsf{end} & (\textit{Inact})
\end{array}
$$

Example: complete purchase

- Roles:
  - Buyer: b, Purchase service: p, credit card service: c
- Private data
  - Credit card number: id ⊗ cc_num
  - Address: id ⊗ address
- Credit card stores: card_store, addr_store

$$
\begin{aligned}
G = {} & \mathtt{b} \to \mathtt{p} : \langle \mathtt{checkout\_request} \rangle . \mathtt{b} \to \mathtt{p} : \langle \mathtt{card\_store} \rangle . \\
& \mathtt{b} \to \mathtt{p} : \langle \mathtt{addr\_store} \rangle . \mathtt{p} \to \mathtt{c} : \langle \mathtt{payment\_request} \rangle . \\
& \mathtt{p} \to \mathtt{c} : \langle \mathtt{card\_store} \rangle . \mathtt{card\_store} \to \mathtt{c} : \langle \_ \otimes \mathtt{cc\_num} \rangle . \\
& \mathtt{c} \to \mathtt{p} : \left\langle \begin{array}{l} \mathtt{auth} : \mathtt{addr\_store} \to \mathtt{p} : \langle \mathtt{id} \otimes \mathtt{address} \rangle . \mathtt{end} \\ \mathtt{deny} : \mathtt{end} \end{array} \right\rangle
\end{aligned}
$$

Local types

$$
\begin{array}{lcll}
T & ::= & \mathsf{p}?\mathsf{U}.T & (\textit{Value Input}) \\
 & | & \mathsf{p}!\mathsf{U}.T & (\textit{Value Output}) \\
 & | & \alpha?[\mathsf{t_i} \otimes \mathsf{g}].T & (\textit{Personal Data Value Input}) \\
 & | & \alpha![\mathsf{t_i} \otimes \mathsf{g}].T & (\textit{Personal Data Value Output}) \\
 & | & \mathsf{p} \oplus \langle \ell_i : T_i \rangle_{i \in I} & (\textit{Selection}) \\
 & | & \mathsf{p}\&\langle \ell_i : T_i \rangle_{i \in I} & (\textit{Branching}) \\
 & | & t \mid \mu t.T & (\textit{Recursion}) \\
 & | & \mathsf{end} & (\textit{Inact})
\end{array}
$$

Projection function

- Given a global type G and a role p, a projection function allows to extract the behavior expected from each role.

$$(\mathsf{p} \to \mathsf{p}'{:}\langle \mathsf{U} \rangle.\, \mathsf{G})\rceil \mathsf{q} = \begin{cases} \mathsf{p}'!\mathsf{U}.\mathsf{G}\rceil \mathsf{q} & \text{if } \mathsf{q} = \mathsf{p}, \\ \mathsf{p}?\mathsf{U}.\mathsf{G}\rceil \mathsf{q} & \text{if } \mathsf{q} = \mathsf{p}', \\ \mathsf{G}\rceil \mathsf{q} & \text{otherwise} \end{cases}$$

$$(\alpha \to \mathsf{p}{:}\langle \mathsf{t_i} \otimes \mathsf{g} \rangle.\, \mathsf{G})\rceil \mathsf{q} = \begin{cases} \alpha?[\mathsf{t_i} \otimes \mathsf{g}].\mathsf{G}\rceil \mathsf{q} & \text{if } \mathsf{q} = \mathsf{p} \\ \mathsf{G}\rceil \mathsf{q} & \text{otherwise} \end{cases}$$

$$(\mathsf{p} \to \alpha{:}\langle \mathsf{t_i} \otimes \mathsf{g} \rangle.\, \mathsf{G})\rceil \mathsf{q} = \begin{cases} \alpha![\mathsf{t_i} \otimes \mathsf{g}].\mathsf{G}\rceil \mathsf{q} & \text{if } \mathsf{q} = \mathsf{p} \\ \mathsf{G}\rceil \mathsf{q} & \text{otherwise} \end{cases}$$

$$(\mathsf{p} \to \mathsf{p}'{:}\langle \{\ell_i : \mathsf{G}_i\}_{i \in I} \rangle)\rceil \mathsf{q} = \begin{cases} \mathsf{p}' \oplus \langle \ell_i : \mathsf{G_i}\rceil \mathsf{q} \rangle_{i \in I} & \text{if } \mathsf{q} = \mathsf{p} \\ \mathsf{p} \& \langle \ell_i : \mathsf{G_i}\rceil \mathsf{q} \rangle_{i \in I} & \text{if } \mathsf{q} = \mathsf{p}' \\ \sqcap i \in I (\mathsf{G}_i\rceil \mathsf{q}) & \text{otherwise} \end{cases}$$

$$(\mu t.\mathsf{G})\rceil \mathsf{q} = \mu t.(\mathsf{G}\rceil \mathsf{q}) \text{ if } \mathsf{q} \in \mathsf{pts}(\mathsf{G}) \qquad (\mu t.\mathsf{G})\rceil \mathsf{q} = \mathsf{end} \text{ if } \mathsf{q} \notin \mathsf{pts}(\mathsf{G})$$

$$t\rceil \mathsf{q} = t \qquad \mathsf{end}\rceil \mathsf{q} = \mathsf{end}$$

Projection to local types

- Example

$$G = \mathtt{b} \to \mathtt{p} : \langle \mathtt{checkout\_request} \rangle . \mathtt{b} \to \mathtt{p} : \langle \mathtt{card\_store} \rangle .$$
$$\mathtt{b} \to \mathtt{p} : \langle \mathtt{addr\_store} \rangle . \mathtt{p} \to \mathtt{c} : \langle \mathtt{payment\_request} \rangle .$$
$$\mathtt{p} \to \mathtt{c} : \langle \mathtt{card\_store} \rangle . \mathtt{card\_store} \to \mathtt{c} : \langle \_ \otimes \mathtt{cc\_num} \rangle .$$
$$\mathtt{c} \to \mathtt{p} : \left\langle \begin{array}{l} \mathtt{auth} : \mathtt{addr\_store} \to \mathtt{p} : \langle \mathsf{id} \otimes \mathtt{address} \rangle . \mathsf{end} \\ \mathtt{deny} : \mathsf{end} \end{array} \right\rangle$$

$$\Downarrow$$

$$G \upharpoonright \mathtt{p} = \mathtt{b}?\mathtt{checkout\_request} . \mathtt{b}?\mathtt{card\_store} . \mathtt{b}?\mathtt{addr\_store} .$$
$$\mathtt{c}!\mathtt{payment\_request} . \mathtt{c}!\mathtt{card\_store} .$$
$$\mathtt{c}\& \left\langle \begin{array}{l} \mathtt{auth} : \mathtt{addr\_store}?(\mathsf{id} \otimes \mathtt{address}) . \mathsf{end}, \\ \mathtt{deny} : \mathsf{end} \end{array} \right\rangle$$

2. Purpose Aware Multi Party Session Types

iii. The modeling Language

The calculus

- The π-calculus [Milner, Parrow, Walker 1992] with sessions [Honda, Vasconcelos, Kubo 1998]
  - Concise yet powerful language for concurrent systems with dynamically-evolving topologies
  - Models systems as a composition of participants, each executing its own process, while communicating with each other by exchanging messages
  - Rich theory in operational, behavioural, and typing semantics
- The Privacy Calculus (Kouzapas and Philippou, 2017)
  - Based on the π-calculus with groups [Cardelli et al., 2000] it introduces the notion of personal data and stores
- Our calculus integrates the personal data and stores from the Privacy Calculus into the π-calculus with sessions
  - r[id⊗data] : r is the address/reference via which access to id⊗data may be provided

The calculus

identities	ι	::=	id $\mid$ _
terms	t	::=	$c \mid$ r $\mid x$
data terms	v	::=	$i \otimes c \mid k$
store terms	u	::=	r $\mid x$
placeholders	k	::=	$x \otimes y \mid \_ \otimes x$
processes	P	::=	$0 \mid \mathrm{p}!\langle t \rangle.P \mid \mathrm{p}?(x).P \mid \mathrm{p} \triangleleft \ell : P \mid \mathrm{p} \triangleright \langle \ell_i : P_i \rangle_{\{i \in I\}}$ $\mid u!\langle v \rangle.P \mid u?(x).P \mid$ if $t = t$ then P else P $\mid X \mid \mu X.P$
networks	M	::=	$\mathrm{p}[P] \mid M \parallel M \mid \mathrm{r}[\mathrm{id} \otimes c]$

Example

$$B = \mathsf{p}!\langle \mathrm{checkout} \rangle.\mathsf{p}!\langle r_1 \rangle.\mathsf{p}!\langle r_2 \rangle.\mathbf{0}$$

$$P = \mathsf{b}?(w).\mathsf{b}?(z_1).\mathsf{b}?(z_2).\mathsf{c}!\langle \mathrm{payment} \rangle.\mathsf{c}!\langle z_1 \rangle.\mathsf{c} \triangleright \left\langle \begin{array}{l} \mathsf{auth} : z_2?(y \otimes x).\mathbf{0}, \\ \mathsf{deny} : \mathbf{0} \end{array} \right\rangle$$

$$C = \mathsf{p}?(w).\mathsf{p}?(z).z?(\_ \otimes x).\mathsf{p} \triangleleft \mathsf{auth}.\mathbf{0}$$

$$M = \mathsf{b} \blacktriangleright [B] \mid \mathsf{p} \blacktriangleright [P] \mid \mathsf{c} \blacktriangleright [C] \mid r_1 \blacktriangleright [\mathsf{id} \otimes \mathrm{cc\_num}] \mid r_2 \blacktriangleright [\mathsf{id} \otimes \mathrm{addr}]$$

Operational semantics

- To reason about behavior of processes, we need to know *how they execute*
- The operational semantics:
  - Specify how communications occur step by step
  - Define process evolution through *reduction rules*
  - Allow formal proofs of key properties
- Each construct of the syntax is associated with rules that describe how respective processes evolve

Operational semantics

- The semantics of the calculus is defined in terms of a structural congruence relation and a reduction relation that specifies how processes evolve

- Reduction for communication:

$$[\text{Comm}]\ \mathsf{p} \blacktriangleright [\mathsf{q}!\langle t\rangle.P_1] \mid \mathsf{q} \blacktriangleright [\mathsf{p}?(x).P_2] \longrightarrow \mathsf{p} \blacktriangleright [P_1] \mid \mathsf{q} \blacktriangleright \left[P_2\{^t/_x\}\right]$$

  - Communication happens when a participant p wishes to sent a value to participant q and the latter is waiting to receive from the former
  - As a result, p proceeds as P_1 and q as P_2 with variable x substituted by t

Semantics

- Reduction for reading from a store

$$[\mathsf{SInp}]\ \mathsf{p} \blacktriangleright [\mathsf{r}?(k).P] \mid \mathsf{r} \blacktriangleright [\mathsf{id} \otimes \mathsf{c}] \longrightarrow \mathsf{p} \blacktriangleright \left[P\{^{\mathsf{id}\otimes\mathsf{c}}/_{k}\}\right] \mid \mathsf{r} \blacktriangleright [\mathsf{id} \otimes \mathsf{c}]$$

- Reduction for writing to a store

$$[\mathsf{SOut}]\ \mathsf{p} \blacktriangleright [\mathsf{r}!\langle \iota \otimes \mathsf{c}\rangle.P] \mid \mathsf{r} \blacktriangleright [\mathsf{id} \otimes \mathsf{c}'] \longrightarrow \mathsf{p} \blacktriangleright [P] \mid \mathsf{r} \blacktriangleright [\mathsf{id} \otimes \mathsf{c}]$$

- Reduction for parallel composition

$$[\mathsf{Par}]\ \frac{M_1 \longrightarrow M}{M_1 \mid M_2 \longrightarrow M \mid M_2}$$

- Reduction under structural congruence

$$[\mathsf{Struct}]\ \frac{M_1 \equiv M_1' \quad M_1' \longrightarrow M_2' \quad M_2' \equiv M_2}{M_1 \longrightarrow M_2}$$

Example

$$
\begin{aligned}
M \longrightarrow\ & \mathsf{b} \blacktriangleright [\mathsf{p}!\langle r_1\rangle.\mathsf{p}!\langle r_2\rangle.\mathbf{0}] \\
& \mid \mathsf{p} \blacktriangleright \left[\mathsf{b}?(z_1).\mathsf{b}?(z_2).\mathsf{c}!\langle \mathrm{payment}\rangle.\mathsf{c}!\langle z_1\rangle.\mathsf{c} \triangleright \left\langle \begin{array}{l} \mathsf{auth} : z_2?(y \otimes x).\mathbf{0}, \\ \mathsf{deny} : \mathbf{0} \end{array} \right\rangle\right] \\
& \mid \mathsf{c} \blacktriangleright [C] \mid r_1 \blacktriangleright [\mathsf{id} \otimes \mathrm{cc\_num}] \mid r_2 \blacktriangleright [\mathsf{id} \otimes \mathrm{addr}] \\
\longrightarrow^*\ & \mathsf{b} \blacktriangleright [\mathbf{0}] \mid \mathsf{p} \blacktriangleright \left[\mathsf{c} \triangleright \left\langle \begin{array}{l} \mathsf{auth} : r_2?(y \otimes x).\mathbf{0}, \\ \mathsf{deny} : \mathbf{0} \end{array} \right\rangle\right] \\
& \mid \mathsf{c} \blacktriangleright [\mathsf{p} \triangleleft \mathsf{auth}.\mathbf{0}] \mid r_1 \blacktriangleright [\mathsf{id} \otimes \mathrm{cc\_num}] \mid r_2 \blacktriangleright [\mathsf{id} \otimes \mathrm{addr}] \\
\longrightarrow\ & \mathsf{b} \blacktriangleright [\mathbf{0}] \mid \mathsf{p} \blacktriangleright [r_2?(y \otimes x).\mathbf{0}] \mid \mathsf{c} \blacktriangleright [\mathbf{0}] \\
& \mid r_1 \blacktriangleright [\mathsf{id} \otimes \mathrm{cc\_num}] \mid r_2 \blacktriangleright [\mathsf{id} \otimes \mathrm{addr}] \\
\longrightarrow\ & \mathsf{b} \blacktriangleright [\mathbf{0}] \mid \mathsf{p} \blacktriangleright [\mathbf{0}] \mid \mathsf{c} \blacktriangleright [\mathbf{0}] \mid r_1 \blacktriangleright [\mathsf{id} \otimes \mathrm{cc\_num}] \mid r_2 \blacktriangleright [\mathsf{id} \otimes \mathrm{addr}]
\end{aligned}
$$

2. Purpose Aware Multi Party Session Types

iv. Typing Purpose Compliance

Policy compliance

- Does a system comply to a purpose?
- Type checking
  - Static technique
  - Validate the proper processing of data inside a system
  - A well-typed system satisfies its purpose protocol
- Our type system uses two type environments:

$$\Gamma ::= \emptyset \mid \Gamma, u : [\mathsf{t_i} \otimes \mathsf{g}]^{\alpha} \mid \Gamma, x : \mathsf{g} \mid \Gamma, X : T$$
$$\Delta ::= \emptyset \mid \Delta, \mathsf{p} : T \mid \Delta, r : [t_i \otimes \mathsf{g}]^{\alpha}$$

  - The type environment Γ maps references u to personal data types of the form $[t_i \otimes g]^{\alpha}$, variables to ground types g, and recursion variables to local types T
  - The session environment Δ maps participants p to local types T and tracks store references

The type system

- Rules for processes have the form $\Gamma \vdash P : T$, meaning that under environment Γ, process P has type T
- They allow us to deduce the local type for a process P
- Example rule:

$$[\mathsf{TOutPD}]\ \frac{\Gamma \vdash u : [\mathsf{t_i} \otimes \mathsf{g}]^{\alpha} \quad \Gamma \vdash c : \mathsf{g} \quad \Gamma \vdash \iota : \mathsf{t_i} \quad \Gamma \vdash P : T}{\Gamma \vdash u!\langle \iota \otimes c \rangle.P : \alpha![\mathsf{t_i} \otimes]\mathsf{g}.T}$$

- Similarly rules for networks allow us to deduce the local types for each participant of a network:

$$[\mathsf{TProc}]\ \frac{\Gamma \vdash P : T}{\Gamma \vdash \mathsf{p} \blacktriangleright [P] \ : \ \mathsf{p}{:}T} \qquad [\mathsf{TPar}]\ \frac{\Gamma \vdash M_1 : \Delta_1 \quad \Gamma \vdash M_2 : \Delta_2}{\Gamma \vdash M_1 \mid M_2 : \Delta_1, \Delta_2}$$

Results: Type Preservation

- We establish that the execution of a well-typed network, according to the operational semantics, preserves the well-typedness of the network.

Theorem 1 (Type Preservation). *Assume a well-typed network M. If M executes a computation step $M \longrightarrow M'$, then the resulting system M' is also well-typed.*

Results: Type Safety

- The type safety lemma states that when a network is well-typed, it does not cause runtime errors, i.e., it cannot reduce to a system containing errors.

Lemma 2 (Type Safety). *Assume a well-typed network M such that $\Gamma \vdash M : \Delta$ for some Γ. Whenever $M \longrightarrow^* M'$, then M' is not an error network.*

Results: Session Fidelity

- The Session Fidelity Theorem states that the execution of a process follows the purpose declared.

Theorem 2 (Session Fidelity). *Assume* $\vdash M : \Delta$. *If* $\Delta \longrightarrow \Delta'$ *then* $\exists \Delta'', M' :$ $M \longrightarrow M'$ *and* $\Delta \longrightarrow \Delta''$ *and* $\vdash M' : \Delta''$.

3. Integrating into the SE cycle

Purpose-aware Diagrams – PA-SDs (1)

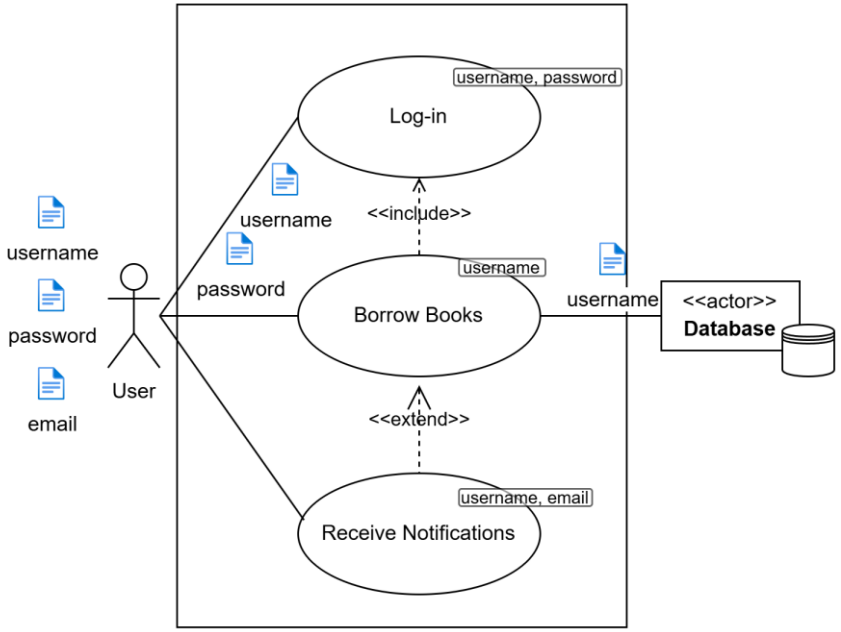
- **Main idea**
  - PA-SDs extend standard UML Sequence Diagrams by explicitly annotating interactions with the personal data involved, allowing the purpose of data processing to be expressed as an ordered sequence of actions rather than a static text label
- **Underlying Aim**
  - Ensure and exploit one-to-one correspondence with PA-MPSTs
- **Step-by-step methodology**
  1. Define functional requirements
  2. Identify system entities
  3. List personal data
  4. Define processing purposes (who provides/reads/processes – which data)
  5. Integrate purposes with requirements using the purpose-aware UML diagrams

Purpose-aware diagrams (1)

- Developers identify personal data, participating actors, and intended actions, producing purpose-aware requirements expressed through Purpose-Aware UML Use Case diagrams.

Use Case diagrams
(high-level overview)

- Personal data stores & ownership
- Data used by each use case
- Input/output personal data displayed clearly



Purpose-aware diagrams (2)

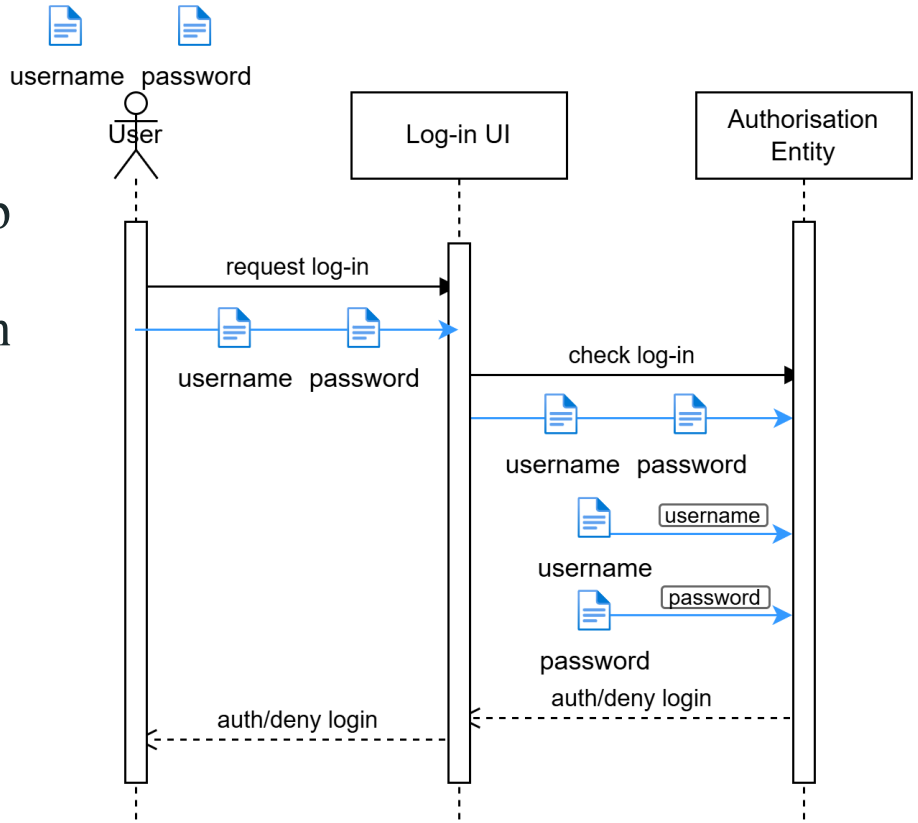
Sequence diagrams

(detailed flows)

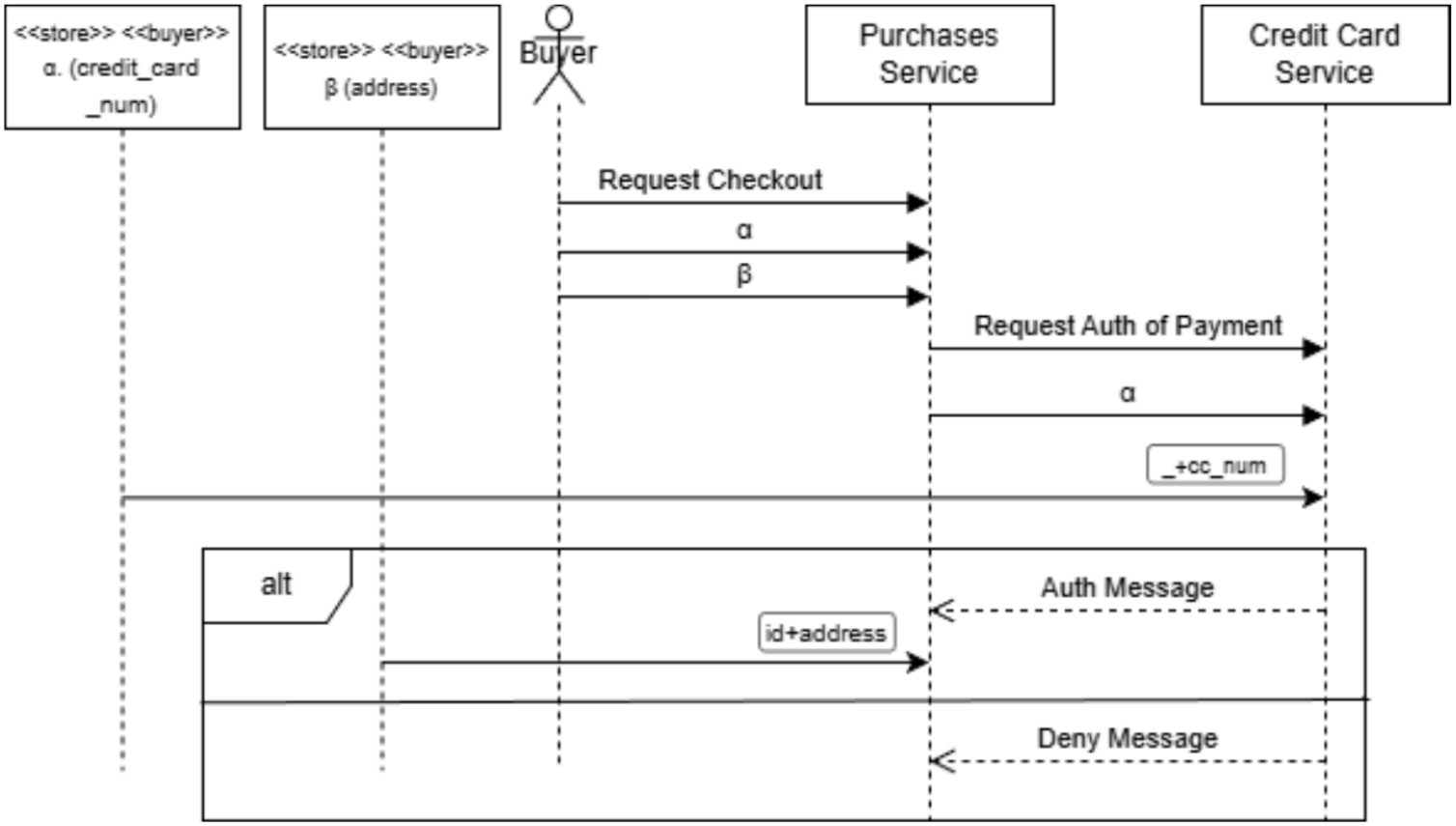
- Personal data stores & ownership
- Personal data exchanges between entities, reading/updating of personal data

Traceable personal-data flows, specifying compliant actions

These sequence diagrams directly correspond to PA-MPSTs

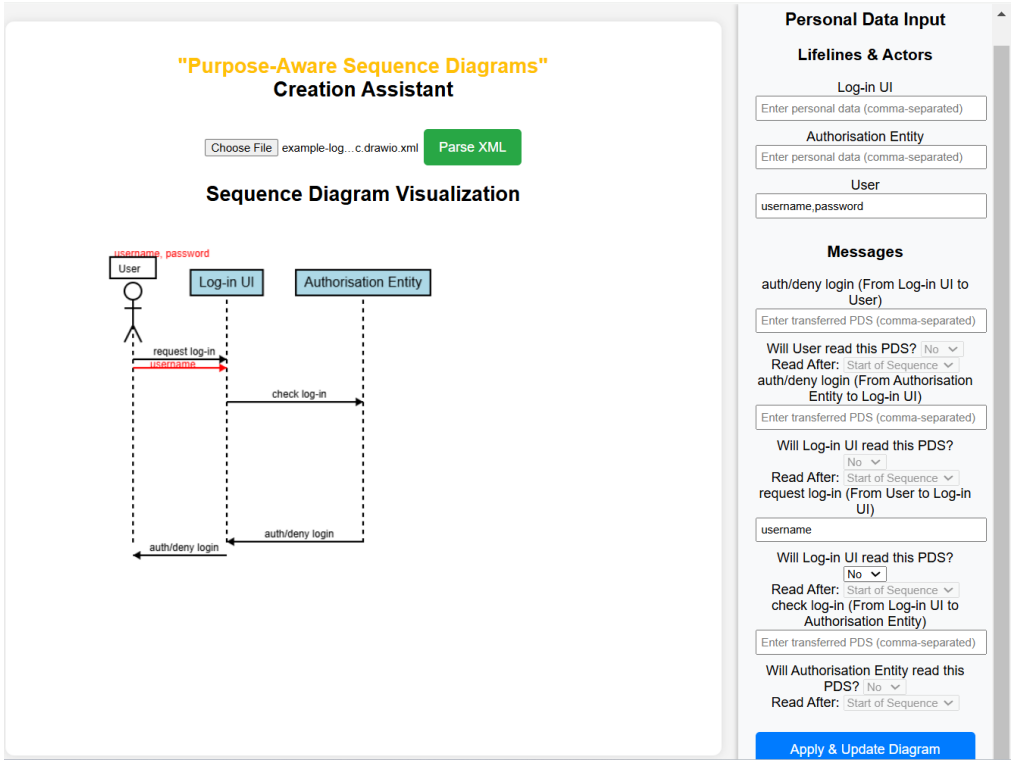


Example

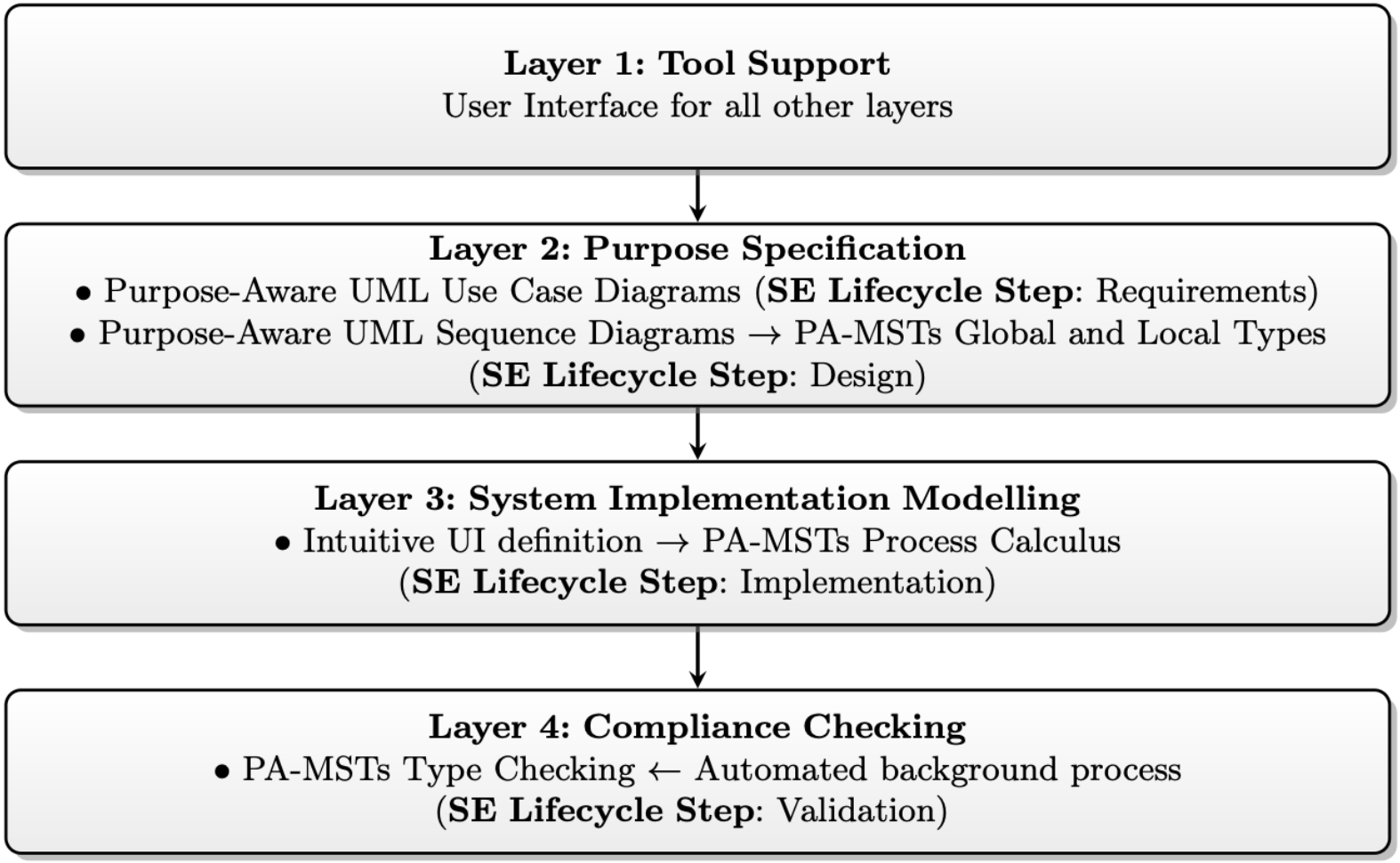

<<store>> <<buyer>>
α. (credit_card
_num)
<<store>> <<buyer>>
β (address)
Buyer
Purchases
Service
Credit Card
Service
Request Checkout
α
β
Request Auth of Payment
α
_+cc_num
alt
Auth Message
id+address
Deny Message

Tool support

- Import a basic sequence diagram (XML)
- Tool auto-detects lifelines, actors, messages, & metadata
- Guides the user to specify data ownership and usage
- Tool generates a purpose-aware sequence diagram + exportable XML



Integrating into the SE cycle – the architecture



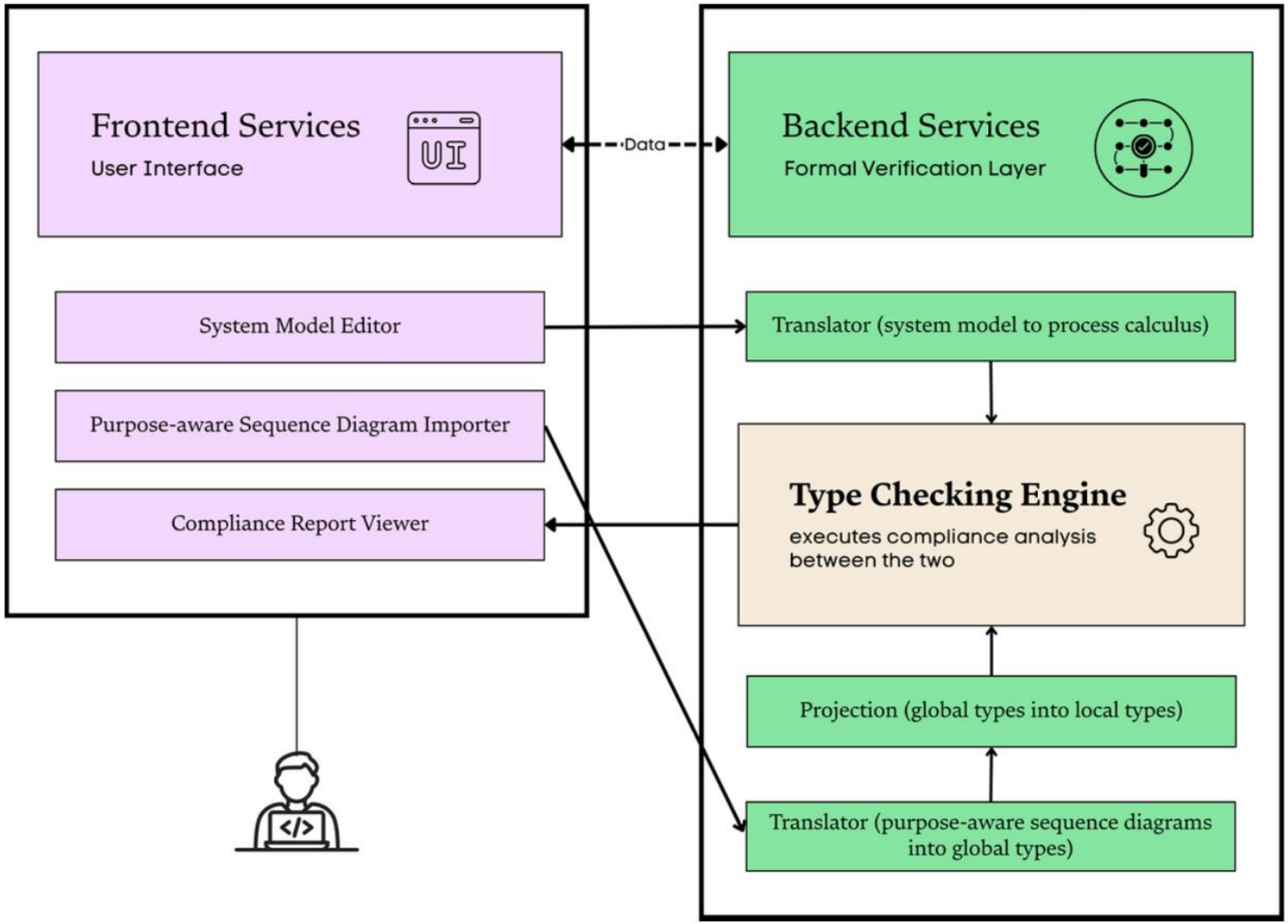
Integrating into the SE cycle – the architecture

- **Tool Support layer:** A graphical user interface enables software engineers to define, visualise, and formally validate purpose-aware system specifications and implementations.
- **Purpose Specification layer:** The processing purposes of the system are defined in terms of actors, personal data elements, and intended actions, through Purpose-Aware UML Use Case and Sequence Diagrams. These are then translated into formal PA-MSTs global and local types.
- **System Implementation/Modelling layer**: The system implementation is modelled via a simple and intuitive UI, and translated into a process calculus models, representing the behaviour of actors and their interactions with personal data stores.
- **Compliance Checking layer:** This layer realises the formal reasoning mechanism that validates conformance between purpose specifications and system implementations through the extended PA-MST type system.

Alignment with the SE cycle

- **Requirements stage**: developers identify personal data, participating actors, and intended actions, producing purpose-aware requirements expressed through Purpose-Aware UML Use Case diagrams
- **Design stage:** requirements are refined and visualised through Purpose-Aware UML SDs, which correspond to global and, consequently, local types in the underlying formal language
- **Implementation stage:** The system implementation models are created using simple visual elements mapped to the formal process calculus, representing executable behaviour between system processes and personal data stores
  - User defines all system actors and personal data stores, and specifies each actor's process via a visual assisting tool, which is then translated into a π-calculus process.
- **Validation stage**: type checking is automatically executed to check purpose compliance between the formal system model and the declared purposes, and a compliance report is produced
  - Validation results can inform revisions to earlier stages, supporting continuous privacy-by-design and iterative refinement.

Tool architecture



Tool architecture

- In the back-end, we have the formal mechanisms:
  - translation of system models from text (received as input from the front-end) into formal process calculus representations;
  - translation of PA-SDs (received as input from the front-end) into global types;
  - projection from global types (received as input from the previous module) into local types; and
  - the type checking engine, receiving as input the formal process calculus model and the local types and producing the compliance report, which is returned to the front-end

4. Conclusions

Future work

- Add support for further analysis of the types and processes and model-checking to validate additional properties of purpose as well as properties relating to other GDPR rights
- Transform formal models into code skeletons/implementations (MDE)
  - Develop tools towards the generation of GDPR-compliant code
  - Type checking code implementations
  - Run-time monitoring

Future work

- Explore scalability on realistic case studies
- Perform an experimental evaluation of the approach among software engineers
- Collaboration between Computer Science and Law
  - Help address ambiguity withing GDPR and address conflicts between different legislations.
  - Access to real privacy scenarios and use-cases
  - Describe the requirements of real use-cases as privacy policies

Thank you! Questions?