Model-driven Privacy

Srđan Krstić

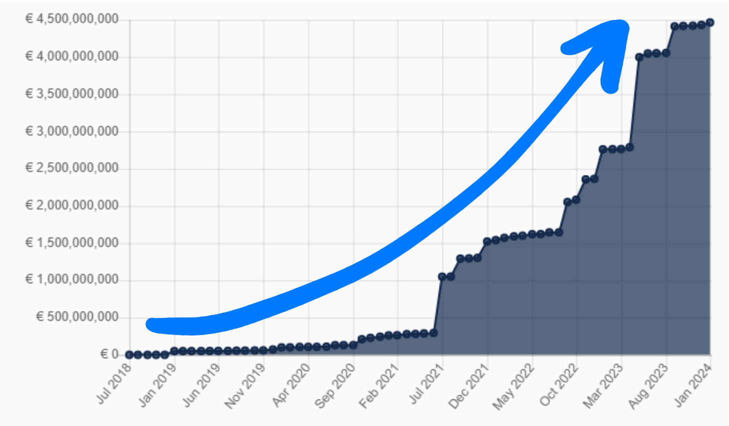
Hoàng Nguyễn

David Basin

Information Security Group
Computer Science Department
ETH Zürich, Switzerland

PETS 24, Bristol, UK

GDPR sum of fines and penalties



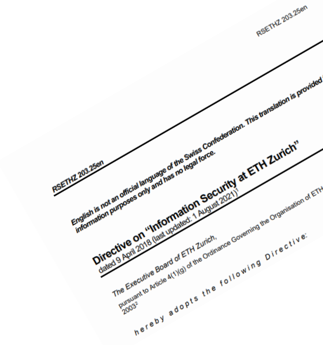
Privacy requirements

Source of requirements:

- Privacy regulations: GDPR, CCPA, DCIA, PIPL
- User preferences and concerns
- Self-imposed organization policies
- Risk-based scenarios and best practices

Common requirements:

- Purpose limitation
- Data subject consent
- Right to rectification, erasure, and restriction
- Data minimization
- Storage limitation
- ...



RSETHZ 203.25en

RSETHZ 203.25en

English is not an official language of the Swiss Confederation. This translation is provided information purposes only and has no legal force.

Directive on "Information Security at ETH Zurich"
dated 9 April 2018 (last updated: 1 August 2021)[1]

The Executive Board of ETH Zurich,

pursuant to Article 4(1)(g) of the Ordinance Governing the Organisation of ETH 2003[2]

hereby adopts the following Directive:

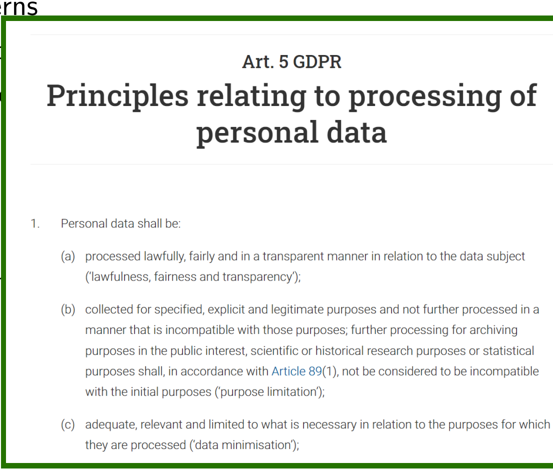
Privacy requirements

Source of requirements:

- Privacy regulations: **GDPR**, CCPA, DCIA, PIPL
- User preferences and concerns
- Self-imposed organization p
- Risk-based scenarios and b

Common requirements:

- **Purpose limitation**
- **Data subject consent**
- Right to rectification, erasu
- Data minimization
- Storage limitation
- ...



Art. 5 GDPR

Principles relating to processing of personal data

1. Personal data shall be:

(a) processed lawfully, fairly and in a transparent manner in relation to the data subject ('lawfulness, fairness and transparency');

(b) collected for specified, explicit and legitimate purposes and not further processed in a manner that is incompatible with those purposes; further processing for archiving purposes in the public interest, scientific or historical research purposes or statistical purposes shall, in accordance with Article 89(1), not be considered to be incompatible with the initial purposes ('purpose limitation');

(c) adequate, relevant and limited to what is necessary in relation to the purposes for which they are processed ('data minimisation');

Personal data shall be collected for specified, explicit and legitimate purposes and not further processed in a manner that is incompatible with those purposes.

(‘Purpose limitation‘) — *Art. 5 §1 (b)*

Processing shall be lawful only if the data subject has given consent to the processing of his or her personal data for one or more specific purposes.

(‘Data subject consent‘) — *Art. 7 §1*

Handling privacy requirements

Current challenges

Specification: Absence of effective languages and tools.

Handling privacy requirements

Current challenges

Specification: Absence of effective languages and tools.

Implementation: Ad hoc, no guarantee of correctness.

Handling privacy requirements

Current challenges

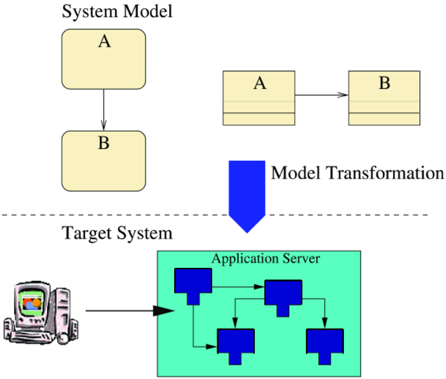
Specification: Absence of effective languages and tools.

Implementation: Ad hoc, no guarantee of correctness.

Evolution/Maintenance: error prone and time consuming.

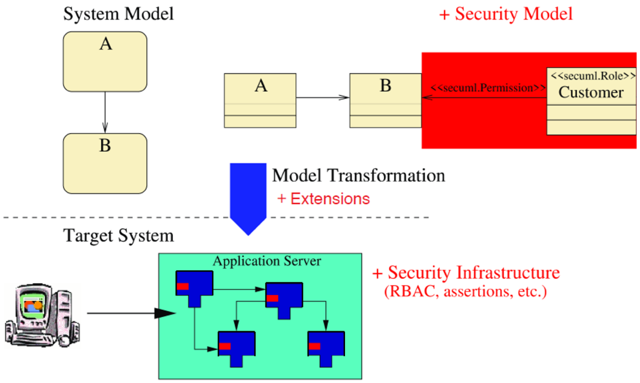
Our solution: **Model-driven development**

Model-driven Development



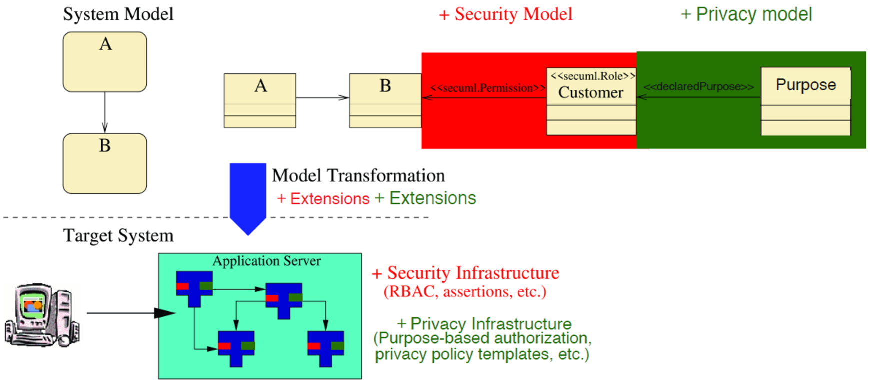
Model-driven Security

Lodderstedt et al. have specialized a new model-driven development methodology that supports **security**.



Model-driven Security and Privacy

Our work: **purpose limitation** and **data subject consent** requirements



Handling privacy requirements

using Model-driven Privacy

✓ **Specification**: Formal language with precise semantics.

Handling privacy requirements

using Model-driven Privacy

✓ **Specification**: Formal language with precise semantics.

✓ **Code generation**: Cross-cutting, correct by design.

Handling privacy requirements

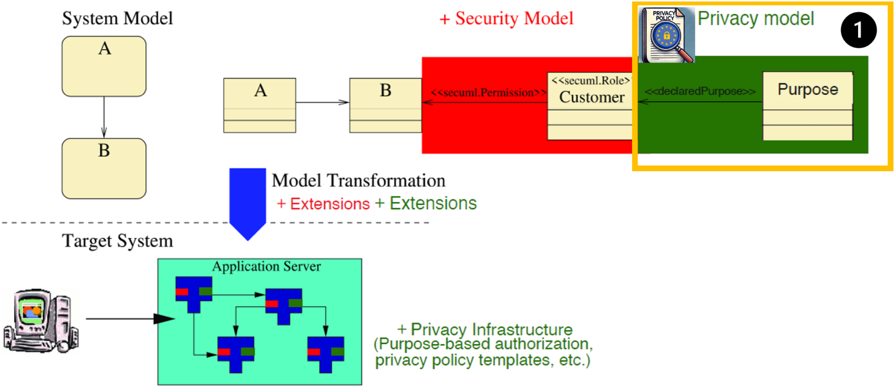
using Model-driven Privacy

✓ **Specification**: Formal language with precise semantics.

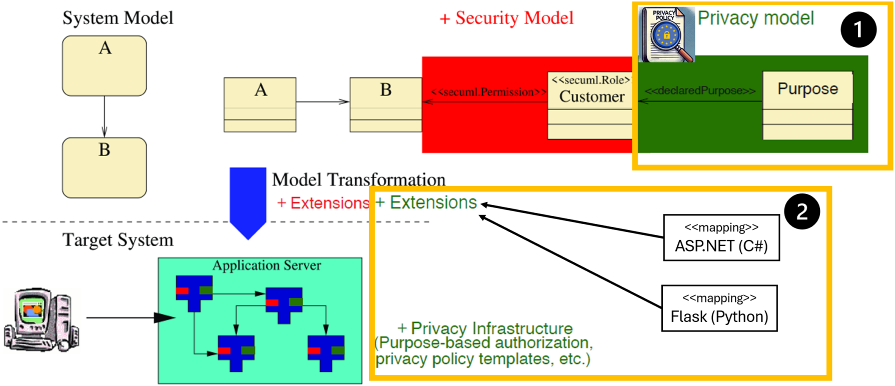
✓ **Code generation**: Cross-cutting, correct by design.

✓ **Evolution**: Change model(s), regenerate code.

Main contribution



Main contribution



System Model

A

B

+ Security Model

A

B

<<secuml.Permission>>

<<secuml.Role>>
Customer

Privacy model

1

<<declaredPurpose>>

Purpose

Model Transformation

+ Extensions

+ Extensions

2

<<mapping>>
ASP.NET (C#)

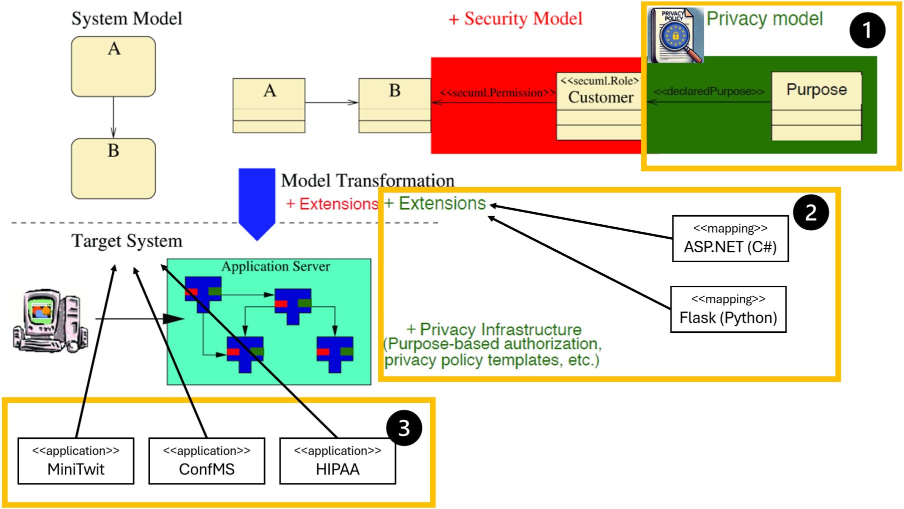
<<mapping>>
Flask (Python)

Target System

Application Server

+ Privacy Infrastructure
(Purpose-based authorization,
privacy policy templates, etc.)

Main contribution



System Model

A

B

+ Security Model

Privacy model

1

A

B

<<secuml.Permission>>

<<secuml.Role>>
Customer

<<declaredPurpose>>

Purpose

Model Transformation

+ Extensions + Extensions

2

<<mapping>>
ASP.NET (C#)

<<mapping>>
Flask (Python)

+ Privacy Infrastructure
(Purpose-based authorization,
privacy policy templates, etc.)

Target System

Application Server

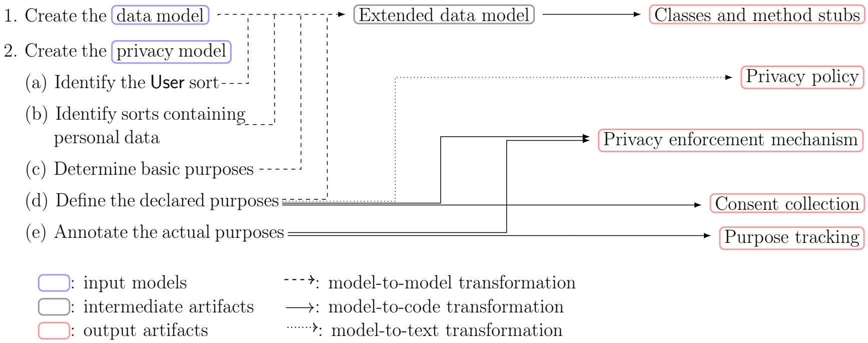
3

<<application>>
MiniTwit

<<application>>
ConfMS

<<application>>
HIPAA

Methodology



1. Create the data model
2. Create the privacy model
   (a) Identify the User sort
   (b) Identify sorts containing personal data
   (c) Determine basic purposes
   (d) Define the declared purposes
   (e) Annotate the actual purposes

Extended data model

Classes and method stubs

Privacy policy

Privacy enforcement mechanism

Consent collection

Purpose tracking

□: input models

□: intermediate artifacts

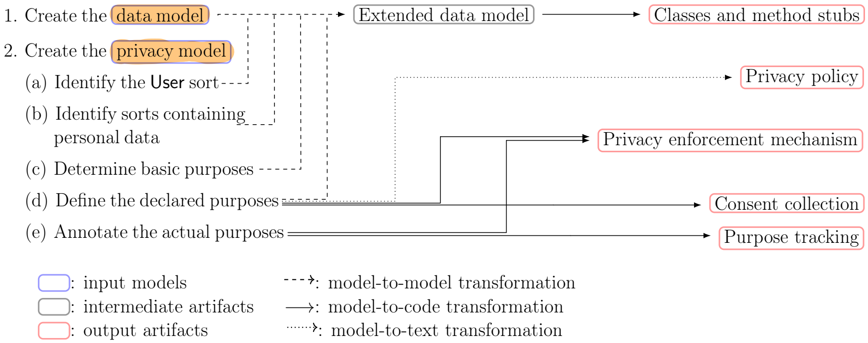
□: output artifacts

⇢: model-to-model transformation

→: model-to-code transformation

⋯→: model-to-text transformation

Methodology



1. Create the data model
2. Create the privacy model
   - (a) Identify the User sort
   - (b) Identify sorts containing personal data
   - (c) Determine basic purposes
   - (d) Define the declared purposes
   - (e) Annotate the actual purposes

Examples

Conference Management System – Data model

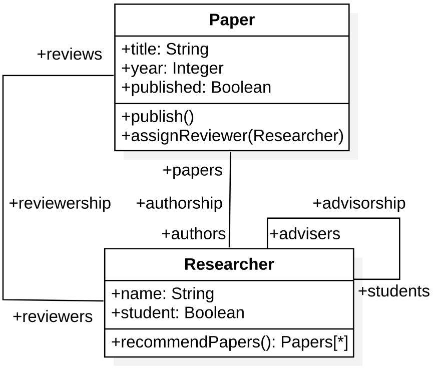
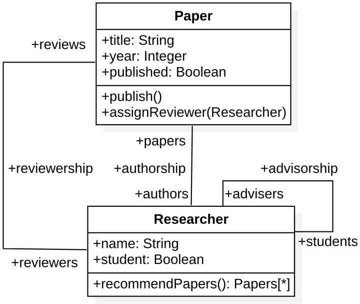


Figure: Data model (UML class diagram)

Examples

Conference Management System – Privacy model

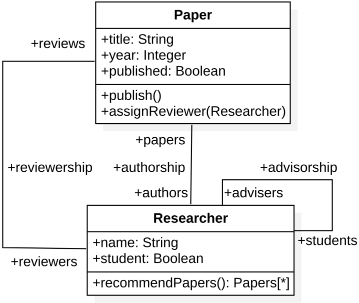
```
{
  "personalData": ["Researcher"], // (2.b.) Identifying personal data
  "purposes": [
    {
      "name": "RecommendPapers",
      "endpoints": [
        {
          "class": "Researcher",
          "met": "recommendPapers"
        }
      ]
    }
  ],
  "policy": [
    {
      "purpose": "RecommendPapers",
      "action": "read",
      "resources": [
        {
          "class": "Researcher",
          "ends": "authors"
        }
      ],
      "constraint": "self.student"
    }
  ]
}
```



Examples

Conference Management System – Privacy model

```
{
  "personalData": ["Researcher"], // (2.b.) Identifying personal data
  "purposes": [ // (2.c) Determining (basic) purposes
    {
      "name": "RecommendPapers",
      "endpoints": [
        {
          "class": "Researcher",
          "met": "recommendPapers"
        }
      ]
    }
  ],
  "policy": [
    {
      "purpose": "RecommendPapers",
      "action": "read",
      "resources": [
        {
          "class": "Researcher",
          "ends": "authors"
        }
      ],
      "constraint": "self.student"
    }
  ]
}
```



Paper

+title: String
+year: Integer
+published: Boolean

+publish()
+assignReviewer(Researcher)

Researcher

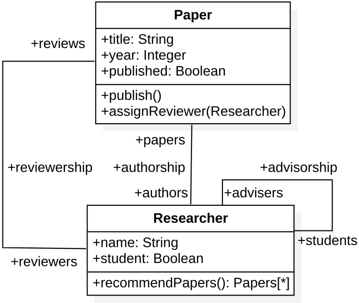
+name: String
+student: Boolean

+recommendPapers(): Papers[*]

Examples

Conference Management System – Privacy model

```
{
  "personalData": ["Researcher"], // (2.b.) Identifying personal data
  "purposes": [ // (2.c) Determining (basic) purposes
    {
      "name": "RecommendPapers",
      "endpoints": [
        {
          "class": "Researcher",
          "met": "recommendPapers"
        }
      ]
    }
  ],
  "policy": [
    { // (2.d.) Defining declared purposes
      "purpose": "RecommendPapers",
      "action": "read",
      "resources": [
        {
          "class": "Researcher",
          "ends": "authors"
        }
      ],
      "constraint": "self.student"
    }
  ]
}
```



Paper

+title: String
+year: Integer
+published: Boolean

+publish()
+assignReviewer(Researcher)

Researcher

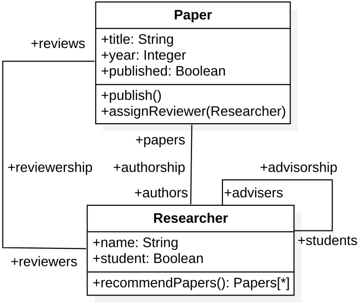
+name: String
+student: Boolean

+recommendPapers(): Papers[*]

Examples

Conference Management System – Privacy model

```
{
  "personalData": ["Researcher"], // (2.b.) Identifying personal data
  "purposes": [ // (2.c) Determining (basic) purposes
    {
      "name": "RecommendPapers",
      "endpoints": [ // (2.e.) Annotating actual purposes
        {
          "class": "Researcher",
          "met": "recommendPapers"
        }
      ]
    }
  ],
  "policy": [
    { // (2.d.) Defining declared purposes
      "purpose": "RecommendPapers",
      "action": "read",
      "resources": [
        {
          "class": "Researcher",
          "ends": "authors"
        }
      ],
      "constraint": "self.student"
    }
  ]
}
```



Paper

+title: String
+year: Integer
+published: Boolean

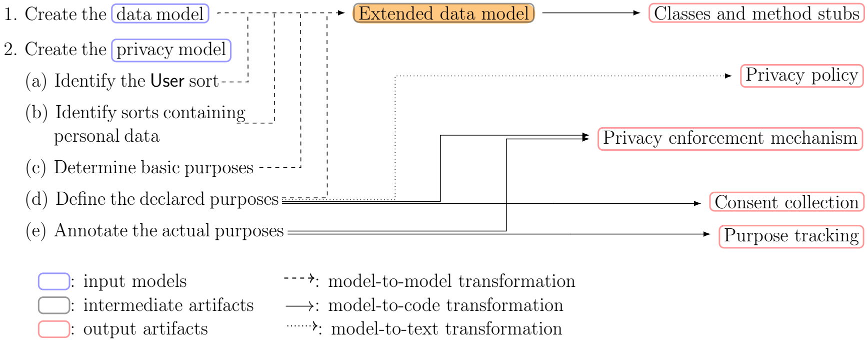
+publish()
+assignReviewer(Researcher)

Researcher

+name: String
+student: Boolean

+recommendPapers(): Papers[*]

Methodology



1. Create the data model
2. Create the privacy model
   (a) Identify the User sort
   (b) Identify sorts containing personal data
   (c) Determine basic purposes
   (d) Define the declared purposes
   (e) Annotate the actual purposes

▭: input models
▭: intermediate artifacts
▭: output artifacts

- - ->: model-to-model transformation
⟶: model-to-code transformation
⋯⋯>: model-to-text transformation

Examples

Conference Management System – Extended data model

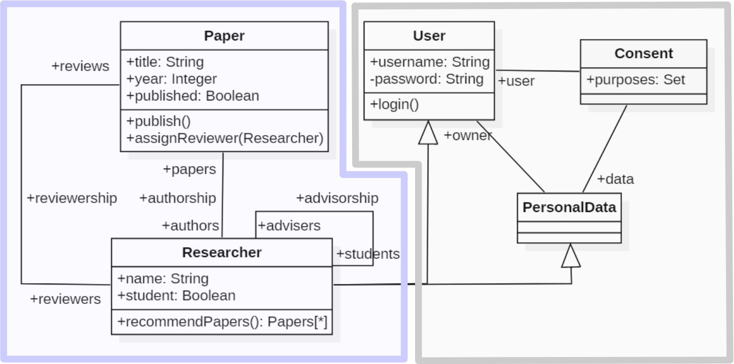
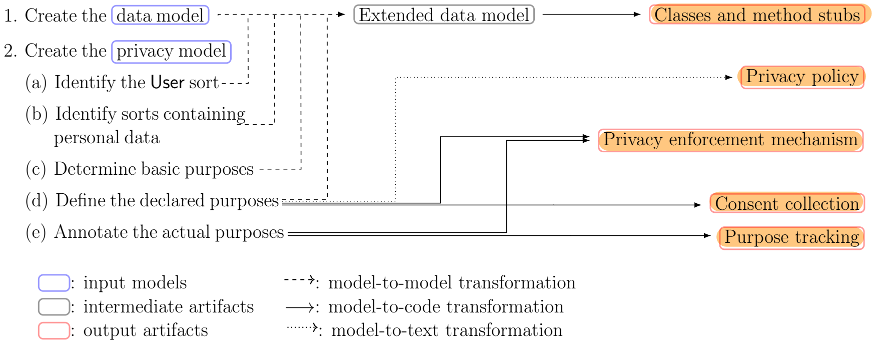


Figure: Data model (extended with privacy classes)

Methodology



1. Create the data model
2. Create the privacy model
   (a) Identify the User sort
   (b) Identify sorts containing personal data
   (c) Determine basic purposes
   (d) Define the declared purposes
   (e) Annotate the actual purposes

Extended data model

Classes and method stubs

Privacy policy

Privacy enforcement mechanism

Consent collection

Purpose tracking

: input models

: intermediate artifacts

: output artifacts

- - -→: model-to-model transformation

⟶: model-to-code transformation

·······→: model-to-text transformation

Examples

Conference Management System – Generated artifacts

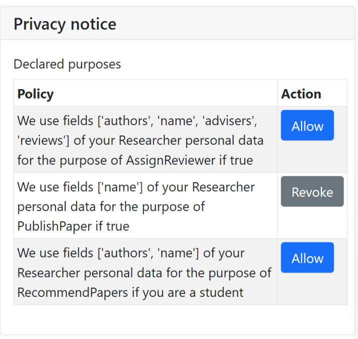
- Methods are generated as empty stubs annotated with their purposes.

```
@label(['RecommendPapers']) // Actual purpose annotation
def recommendPapers():
    // TODO: Implement method stub
```

Examples

Conference Management System – Generated artifacts

- Privacy notice is generated automatically.



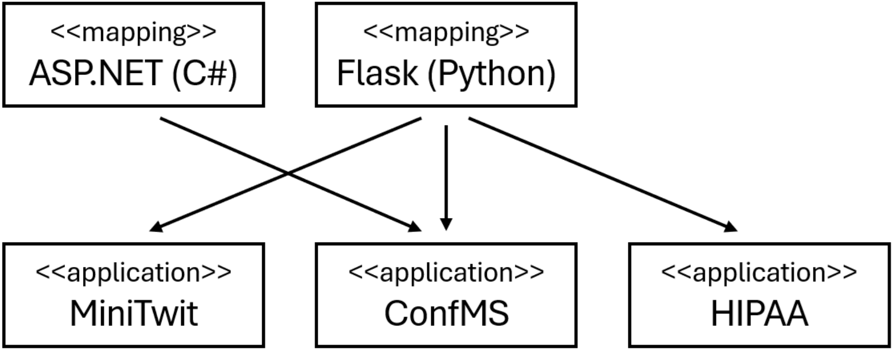
Privacy notice

Declared purposes

Policy	Action
We use fields ['authors', 'name', 'advisers', 'reviews'] of your Researcher personal data for the purpose of AssignReviewer if true	Allow
We use fields ['name'] of your Researcher personal data for the purpose of PublishPaper if true	Revoke
We use fields ['authors', 'name'] of your Researcher personal data for the purpose of RecommendPapers if you are a student	Allow

Implementation

Model transformations:



Case study applications:

Evaluation

(Selected[1]) Research questions

- **Development effort**
  How much developer effort is required to use our approach?

- **Performance overhead**
  How much runtime overhead does our approach incur?

[1] More in the paper

Evaluation

Development effort (on Conference Management System case study)

- Set up: Define models → Generate code → Implement methods.

Evaluation

Development effort (on Conference Management System case study)

- Set up: Define models → Generate code → Implement methods.
- Define models:
  - 13 LoC (data model)
  - 20 LoC (security + privacy model).

Evaluation

Development effort (on Conference Management System case study)

- Set up: Define models → Generate code → Implement methods.
- Define models:
  - 13 LoC (data model)
  - 20 LoC (security + privacy model).
- Generate code:
  - 1954 LoC (C#)
  - 731 LoC (Python)

Evaluation

Development effort (on Conference Management System case study)

- Set up: Define models → Generate code → Implement methods.
- Define models:
  - 13 LoC (data model)
  - 20 LoC (security + privacy model).
- Generate code:
  - 1954 LoC (C#)
  - 731 LoC (Python)
- Implement methods:
  - 344 LoC (C#)
  - 142 LoC (Python)

Evaluation

Development effort (on Conference Management System case study)

- Set up: Define models → Generate code → Implement methods.
- Specification: 33 LoC
- Implementation:
  - 2298 LoC (C#, 85% generated)
  - 873 LoC (Python, 84% generated)

Developers need to implement **only** 15-16% of the overall codebase.

Evaluation

Performance overhead (on MiniTwit case study)

- open-source, unsecured application (baseline)

Evaluation

Performance overhead (on MiniTwit case study)

- open-source, unsecured application (baseline)
  - manually implement privacy checks (secured)

Evaluation

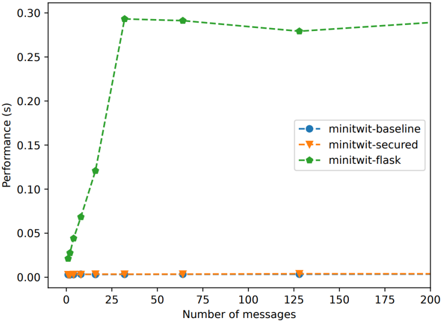
Performance overhead (on MiniTwit case study)

- open-source, unsecured application (baseline)
  - manually implement privacy checks (secured)
  - implement application our approach (flask)

Evaluation

Performance overhead (on MiniTwit case study)

- open-source, unsecured application (baseline)
  - manually implement privacy checks (secured)
  - implement application our approach (flask)
- execute `public_timeline()` endpoint (pagination for 30 messages).



Performance overhead is modest compared to manual implementation.

Future Work

- extend to other class of privacy requirements.

Future Work

- extend to other class of privacy requirements.
- proving the correctness of the transformation.

Future Work

- extend to other class of privacy requirements.
- proving the correctness of the transformation.
- conduct a user (i.e., developer) case study.

Questions?

"Before I write my name on the board, I'll need to know how you're planning to use that data."