An Approach to Automatically Detect Problems in Restructured Deployment Models based on Formalizing Architecture and Design Patterns

University of Stuttgart
Universitätsstr. 38
70569 Stuttgart
Germany

Karoline Saatkamp, Uwe Breitenbücher,
Oliver Kopp, and Frank Leymann
Institute of Architecture of Application Systems
[lastname]@iaas.uni-stuttgart.de

Phone +49 711 685 88 476
Fax +49 711 685 88 472

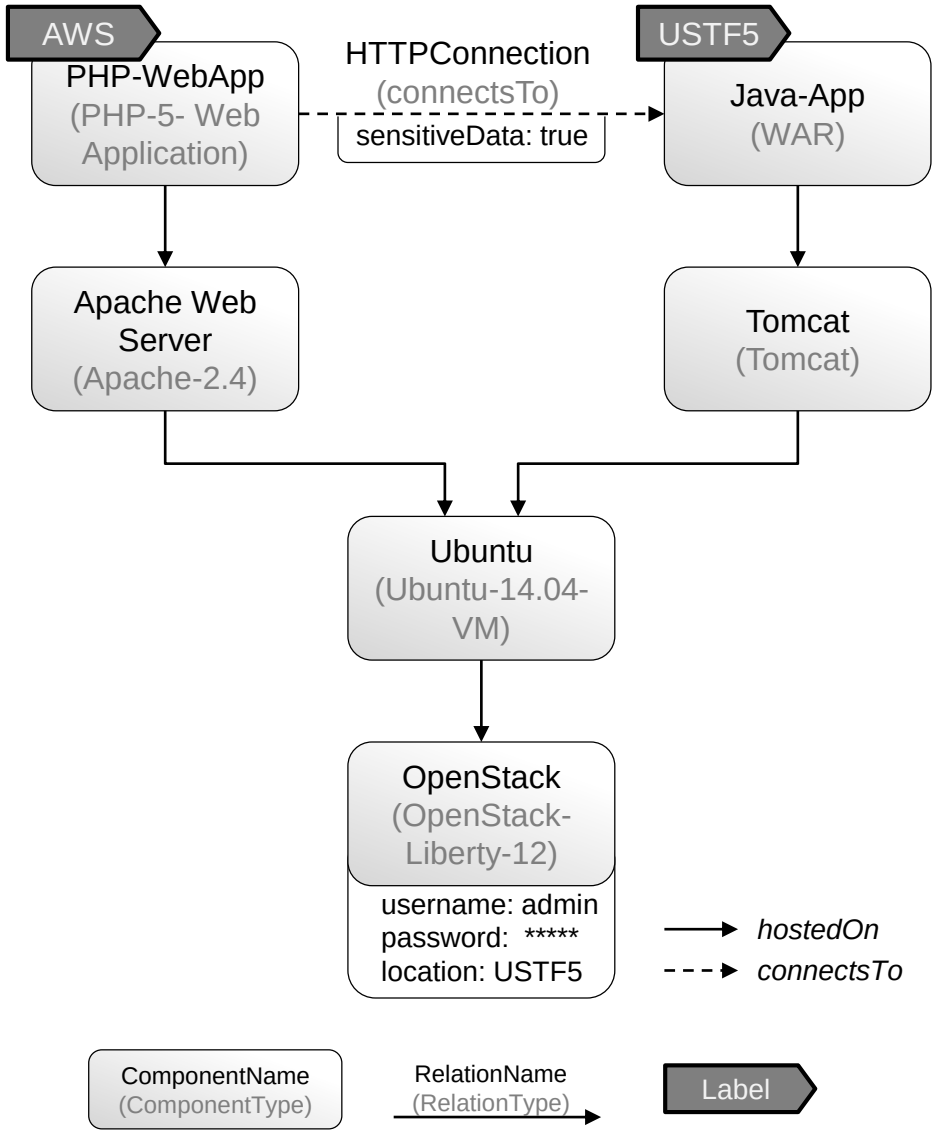
Outline

- Motivation
- Appyling Architecture & Design Knowledge to Deployment Models
- Automated Problem Detection by Formalized Patterns
- Application & Conclusion

Motivation

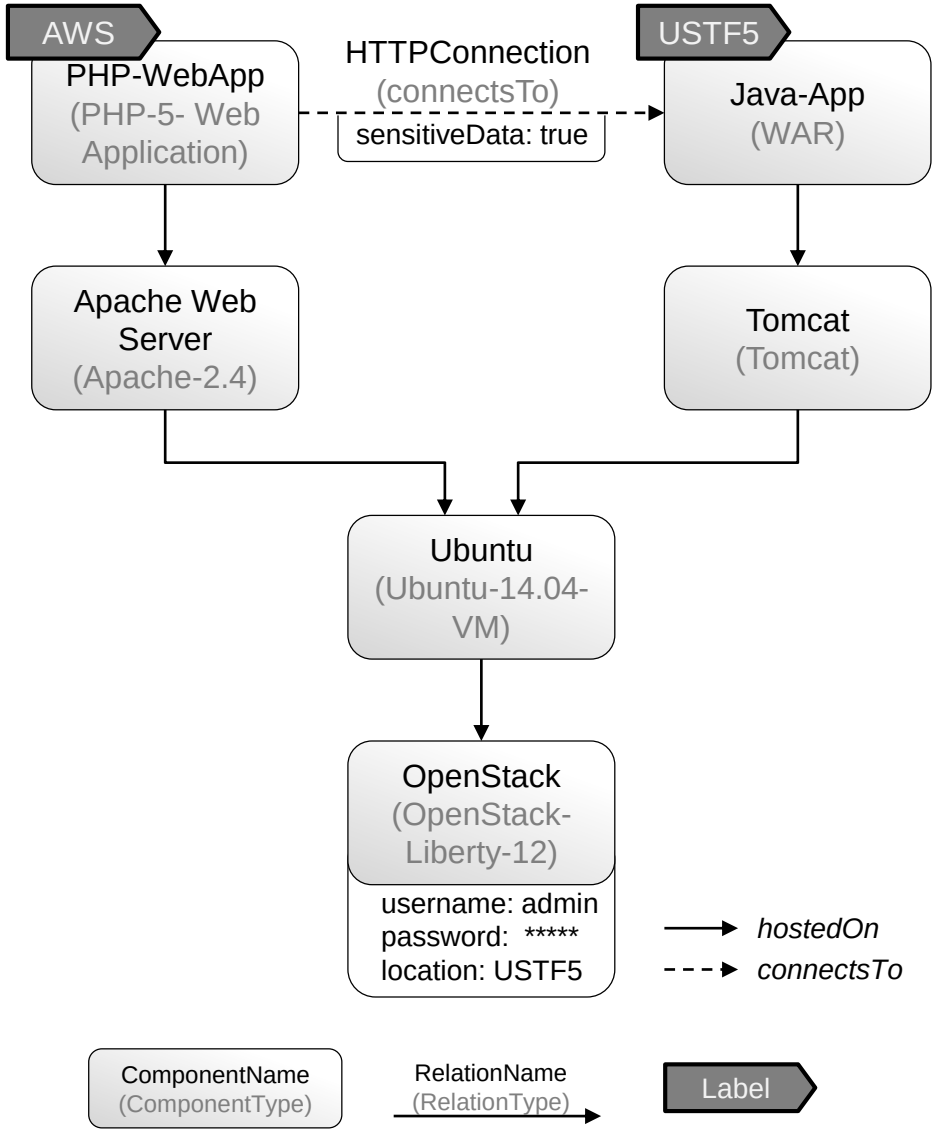
What is a restructured deployment model?

- Deployment Model => Topology-based Deployment Model
- Topology describing the structure of an application for the deployment:
  - Components
  - Relations
  - [Component/Relation] Types
  - Properties
- **Restructuring** means that the components are redistributed across different target locations
- How to model the desired restructuring? → Target Labels*

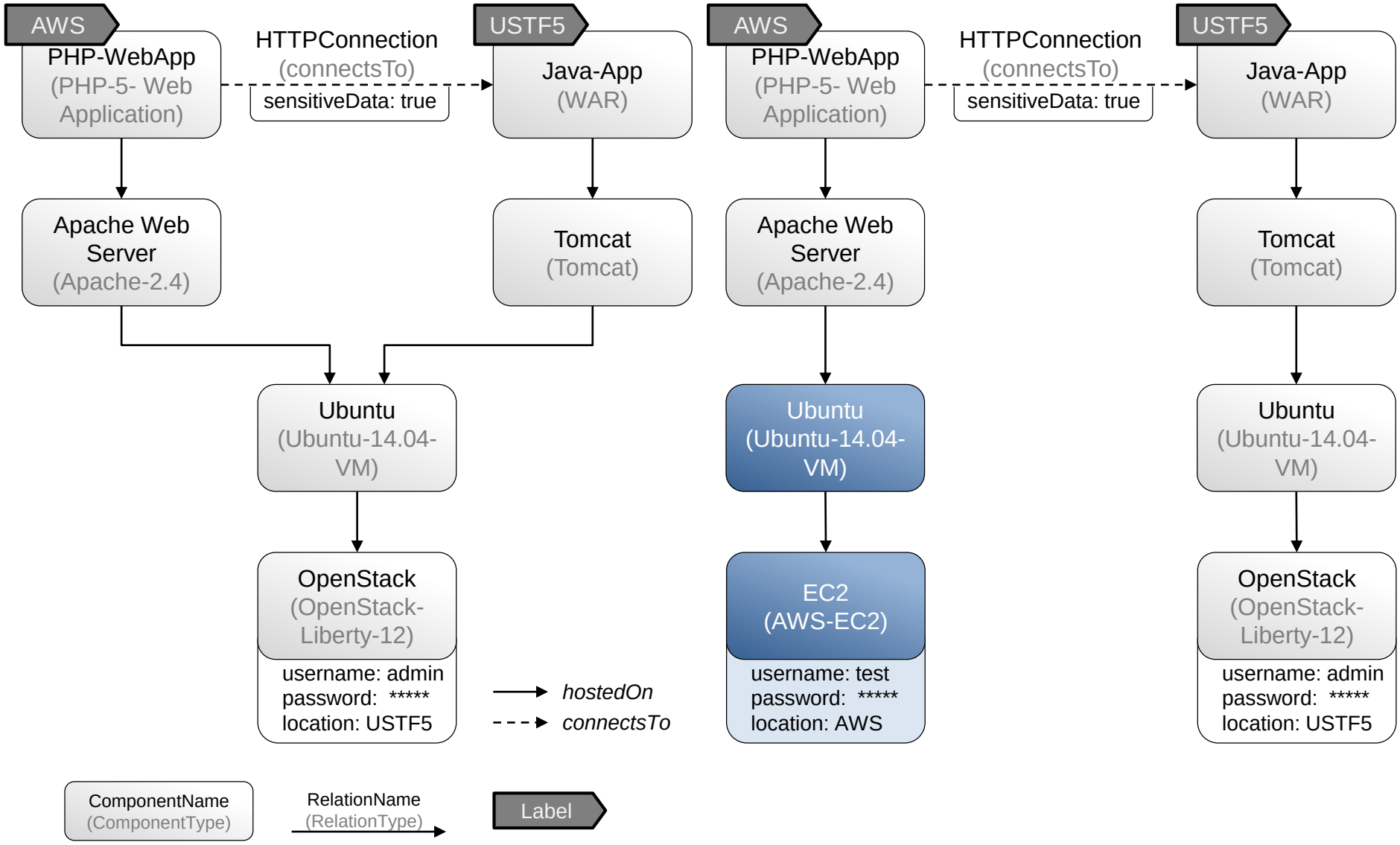


*K. Saatkamp, U. Breitenbücher, O. Kopp, F. Leymann (2017) Topology Splitting and Matching for Multi-Cloud Deployments. In: CLOSER 2017, pp 247-258

What is a restructured deployment model?



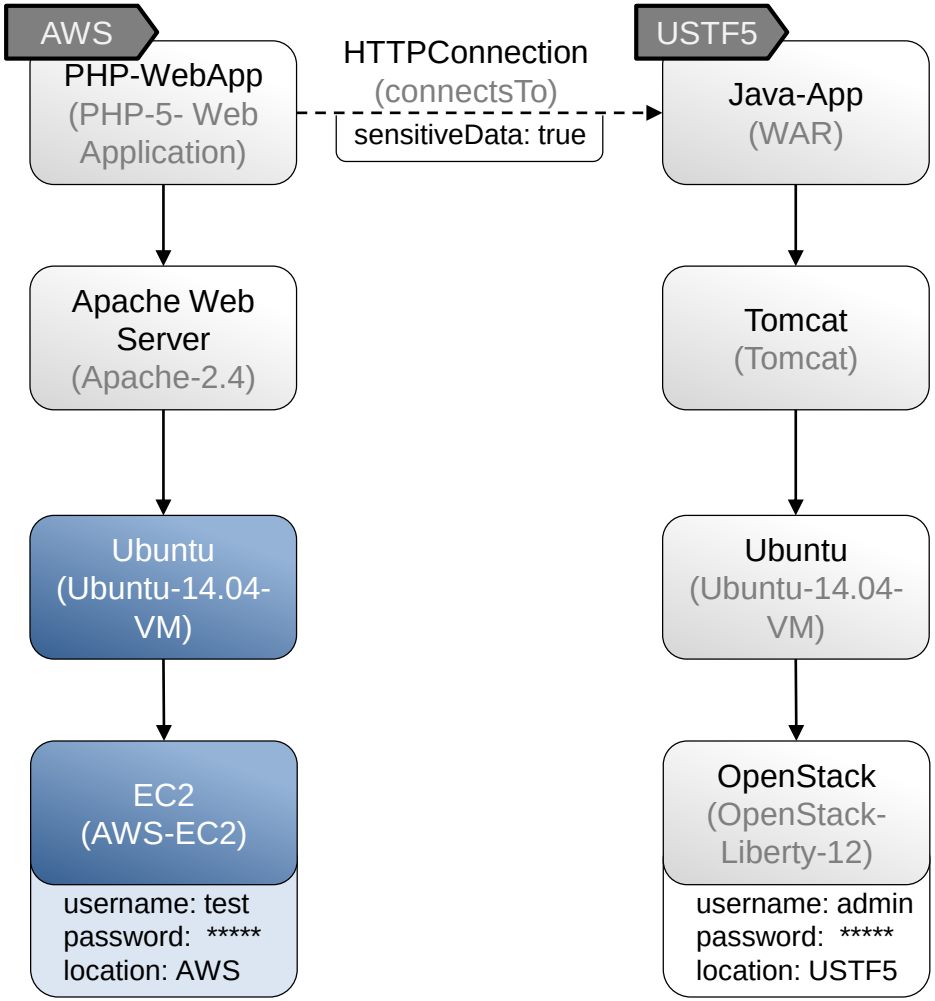
What is a restructured deployment model?



What are the consequences of a restructuring?

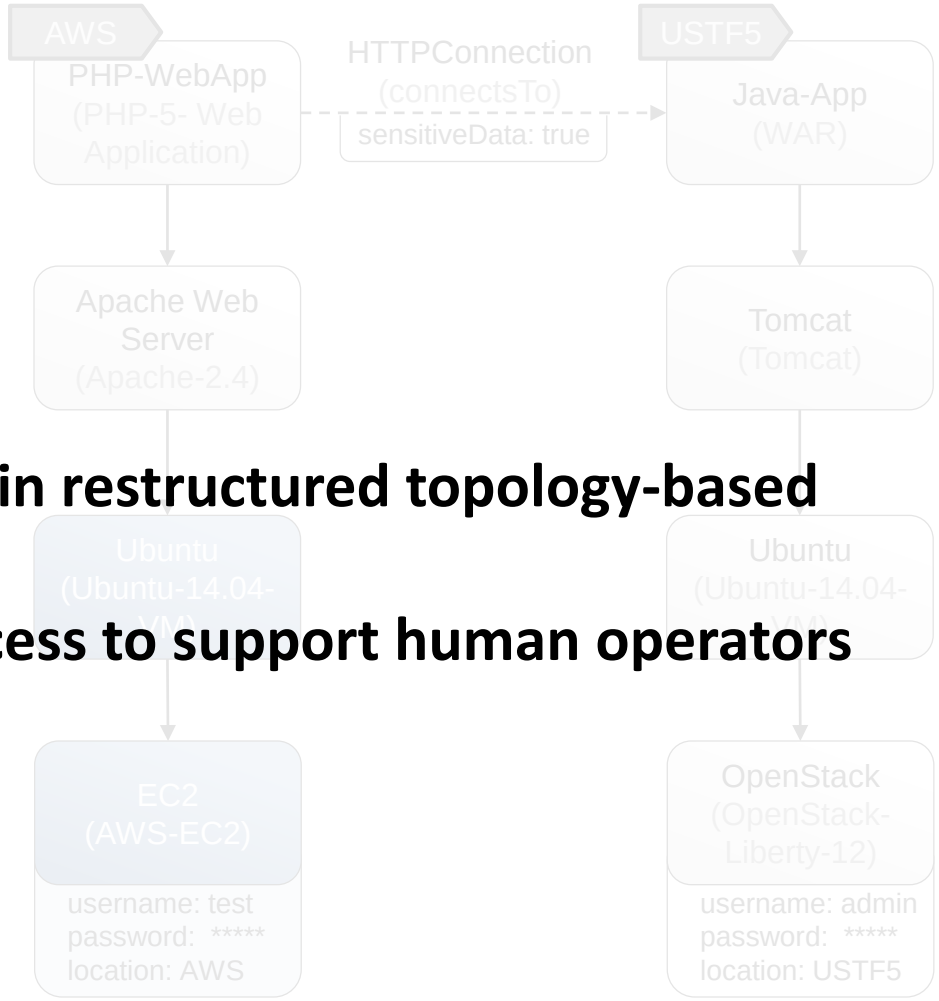
- Components formally hosted on the same host are located in different cloud environments
- The communication tel
  betwee
- [illegible] data are
  [illegible]xchanged over a public network
- Further consequences could be:
  - Firewalls
  - Different Communication Infrastructure
  - External Data Access
  - ...

Do we have a problem?



What are the consequences of a restructuring?

- Components formally hosted on the same host are located in different cloud environments



Thus, we want to …

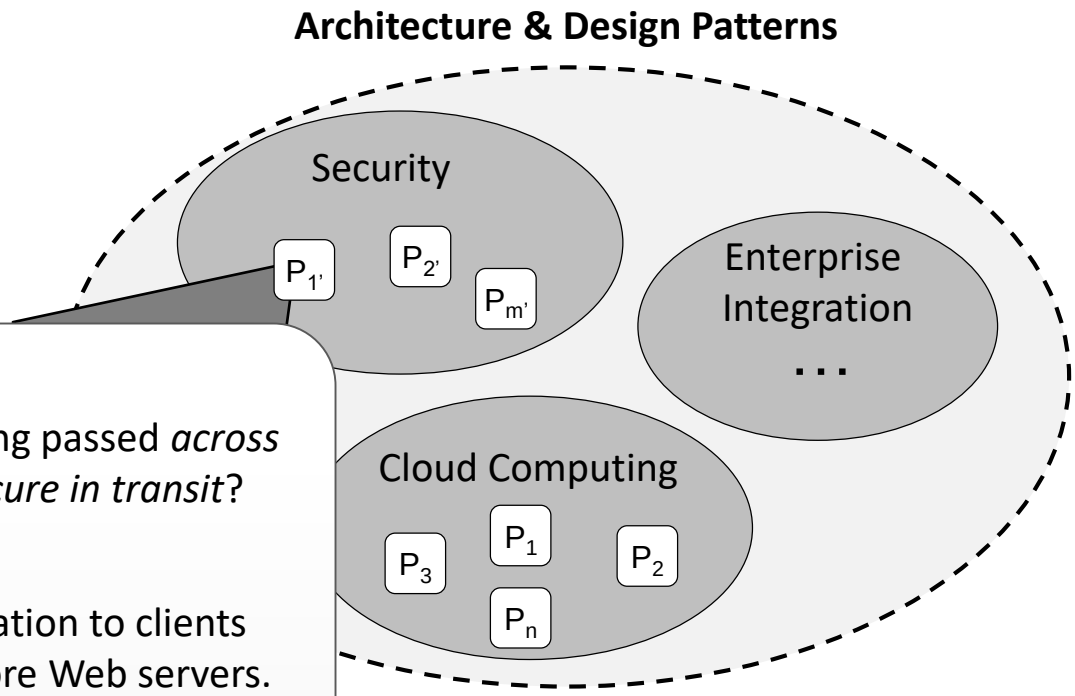
(i) Enable the problem detection in restructured topology-based deployment models and
(ii) Automated this detection process to support human operators

- Further consequences could be:
  - Firewalls
  - Different Communication Infrastructure
  - External Data Access
  - …

Applying Architecture & Design Knowledge to Deployment Models

Architecture & Design Knowledge as Patterns

- Patterns describe design & architectural knowledge as **best practices for recurring problems**



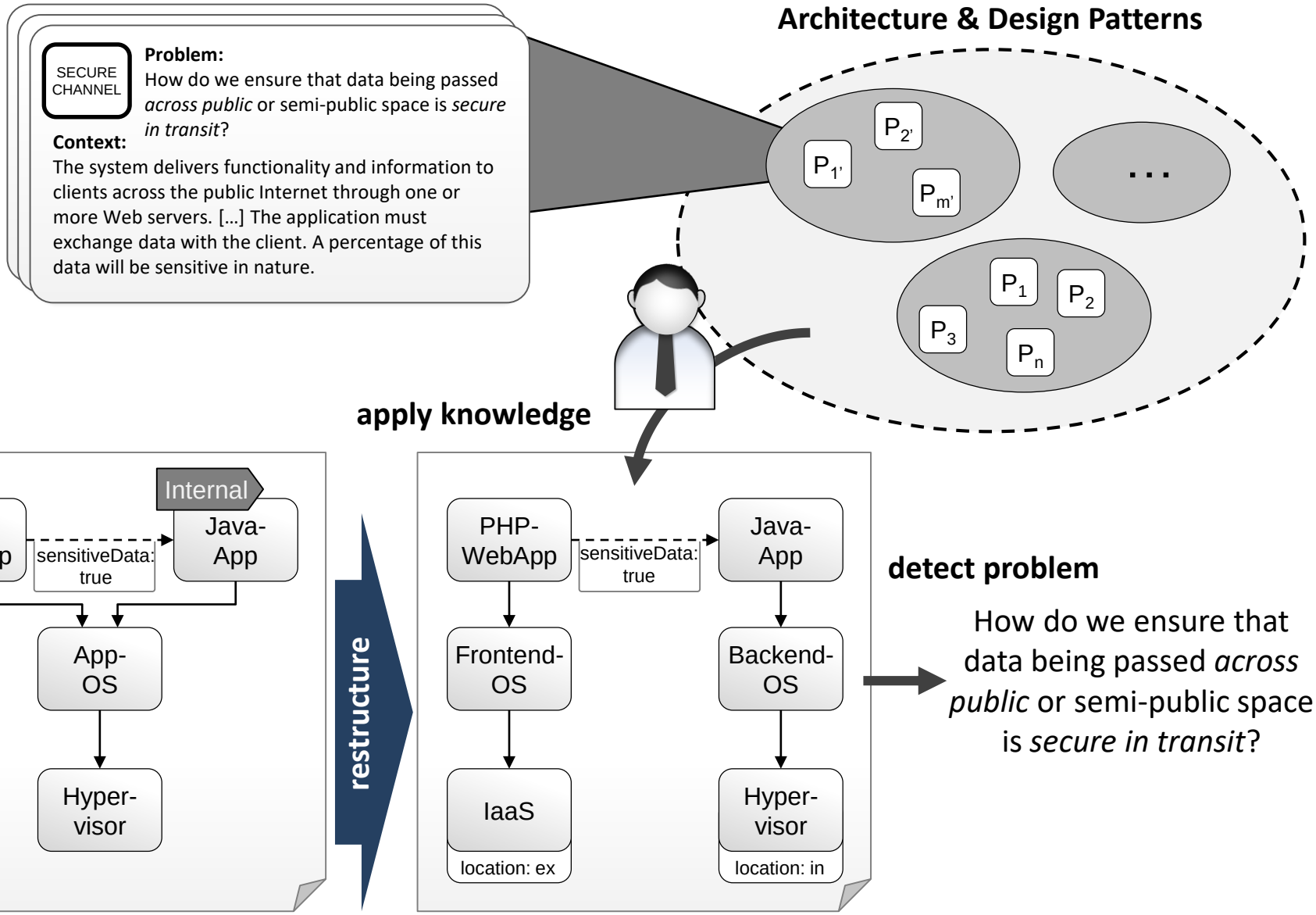
SECURE CHANNEL

Problem:
How do we ensure that data being passed *across public* or semi-public space is *secure in transit*?

Context:
The system delivers functionality and information to clients across the public Internet through one or more Web servers. […] The application must exchange data with the client. A percentage of this data will be sensitive in nature.

Solution: Create secure channels for sensitive data that obscure the data in transit. Exchange information between client and server to allow them to set up encrypted communication between themselves.[…]

Problem Detection Approach



Automated Problem Detection by Formalized Patterns

Pattern Formalization based on Logic Programming

- **Logic Programming:** domain knowledge can be expressed as **facts** and **rules** and it can be checked whether a fact satisfies the conditions of a rule.
  - **Facts** describe specific circumstances, i.e., objects and their relationships.
  - **Rules** simplify querying the facts
- Approach based on the widely used logic programming language **Prolog**

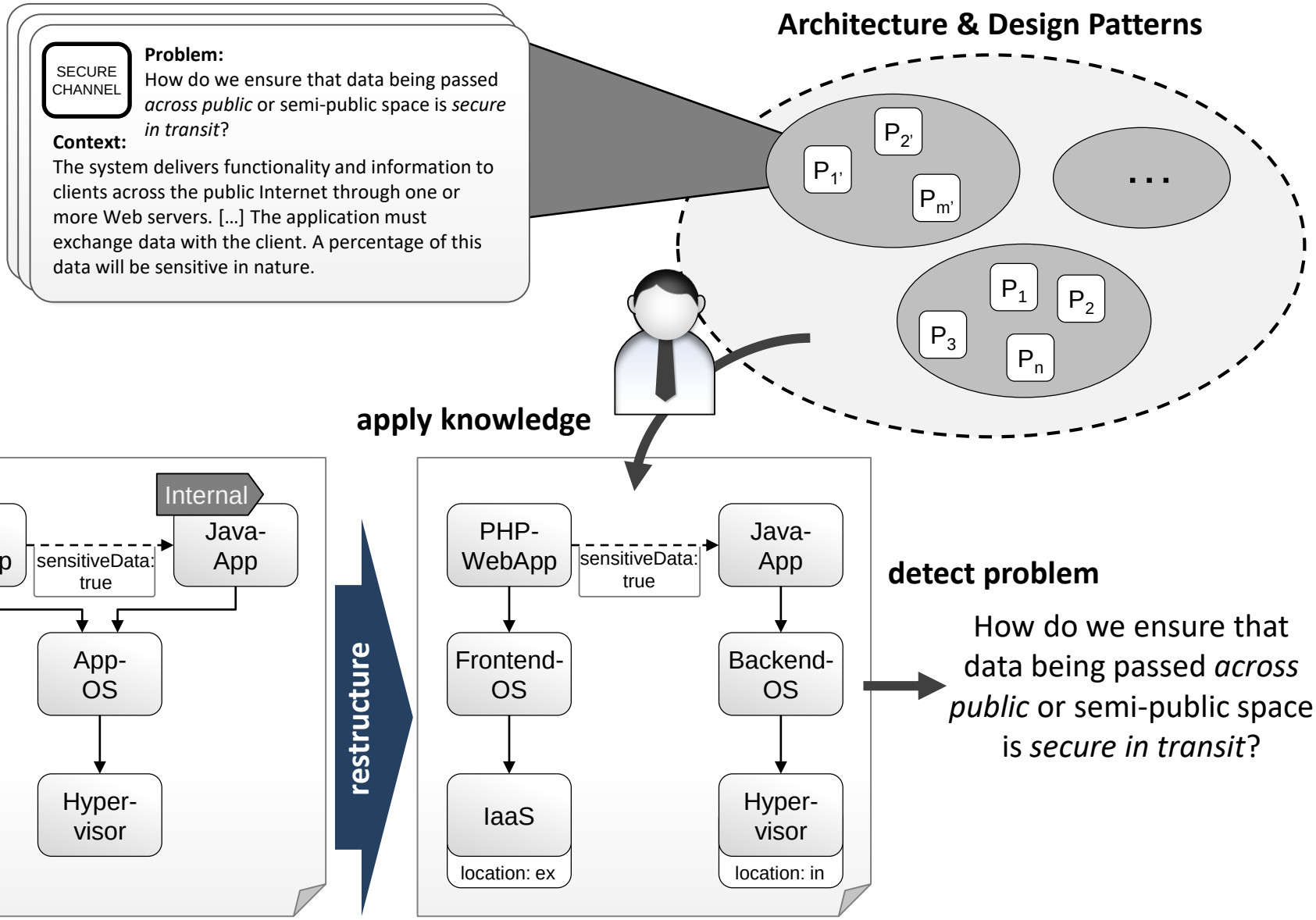
```
father(ben,john).
male(ben).
male(john).
```

```
son(X,Y) :-
  father(Y,X),
  male(Y),
  male(X).
```

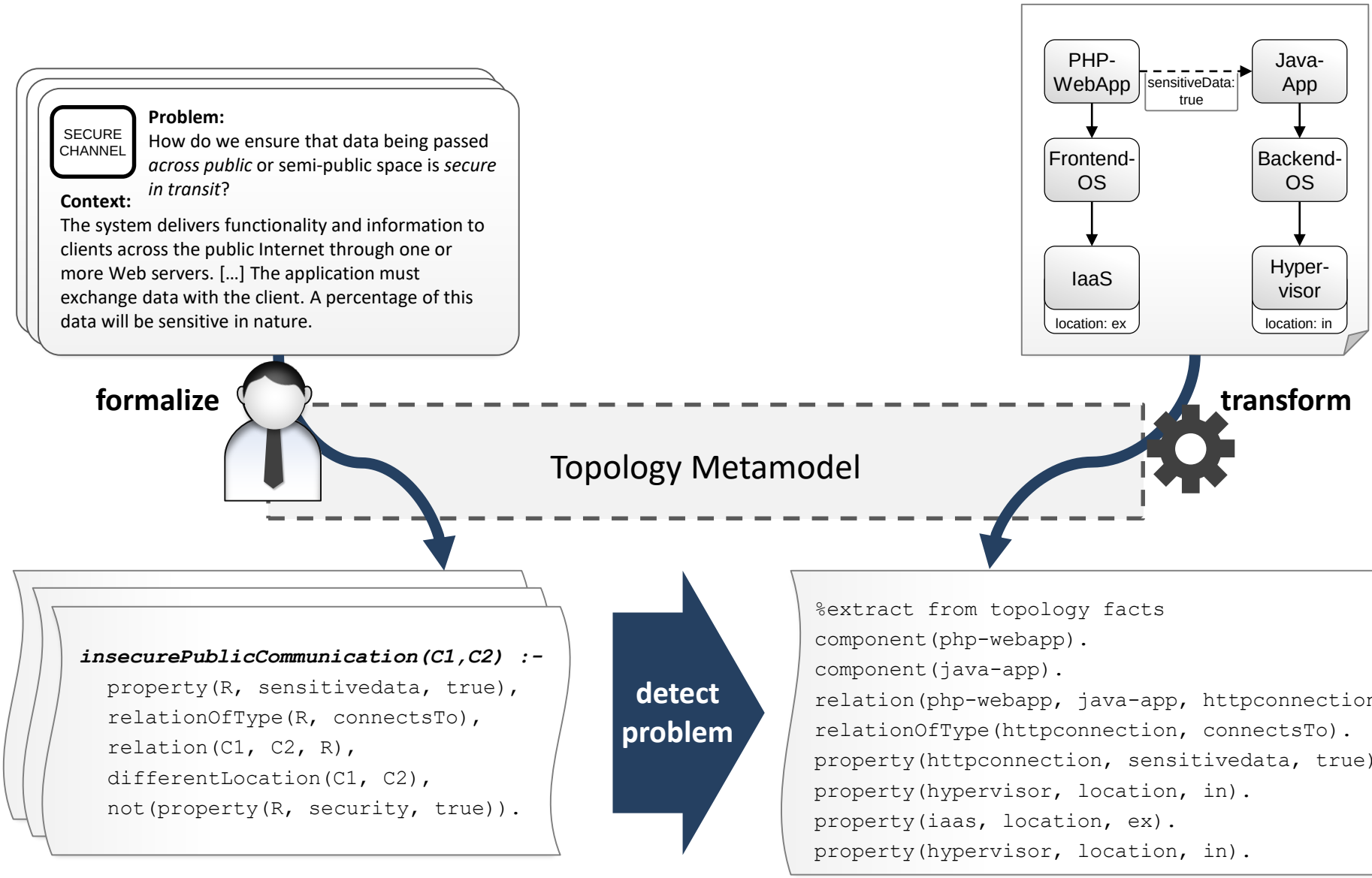
Is John the son of Ben? → YES

Do objects exists for which son(X,Y) is fulfilled? → X= john, Y= ben

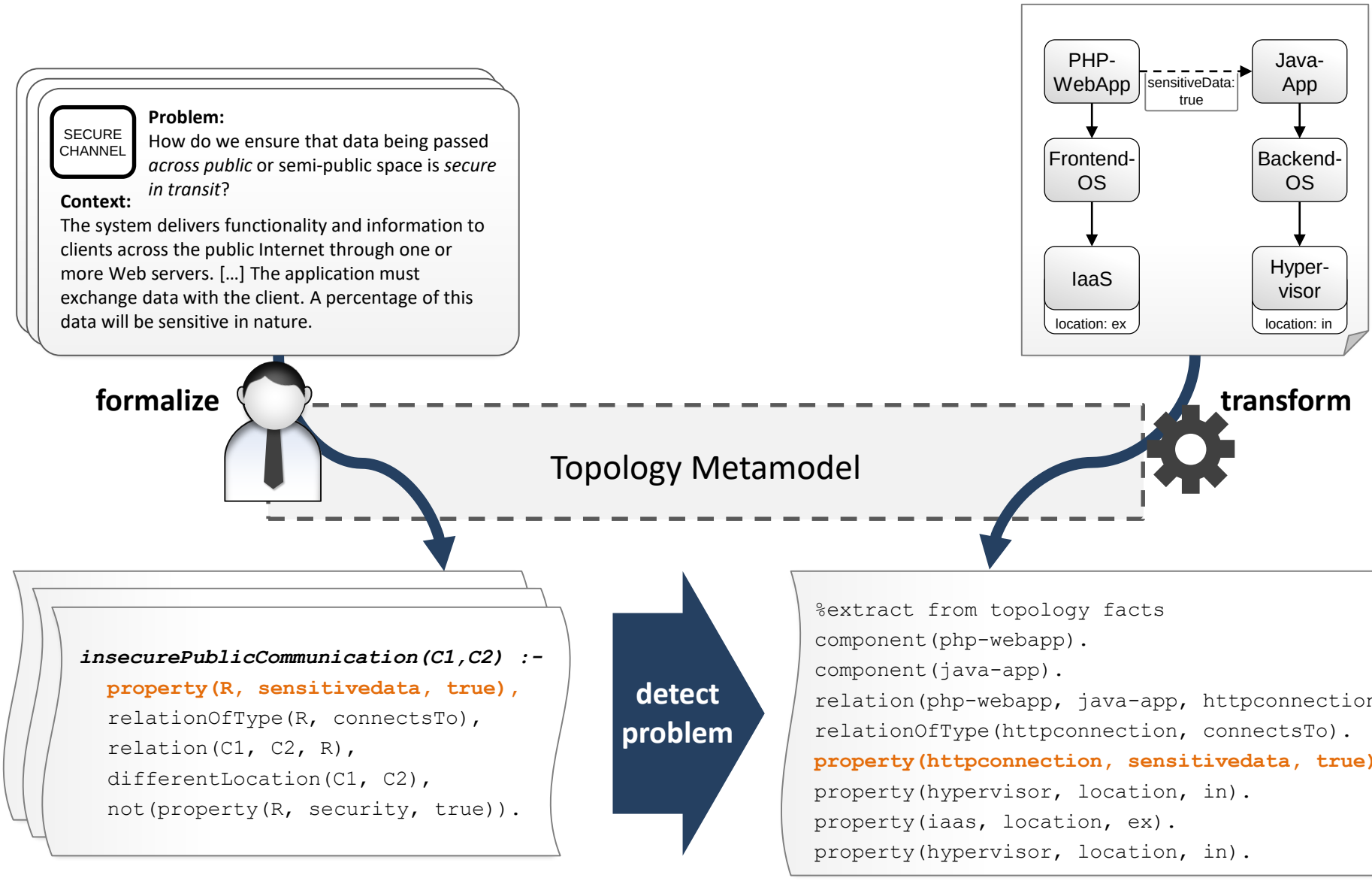
Problem Detection Approach



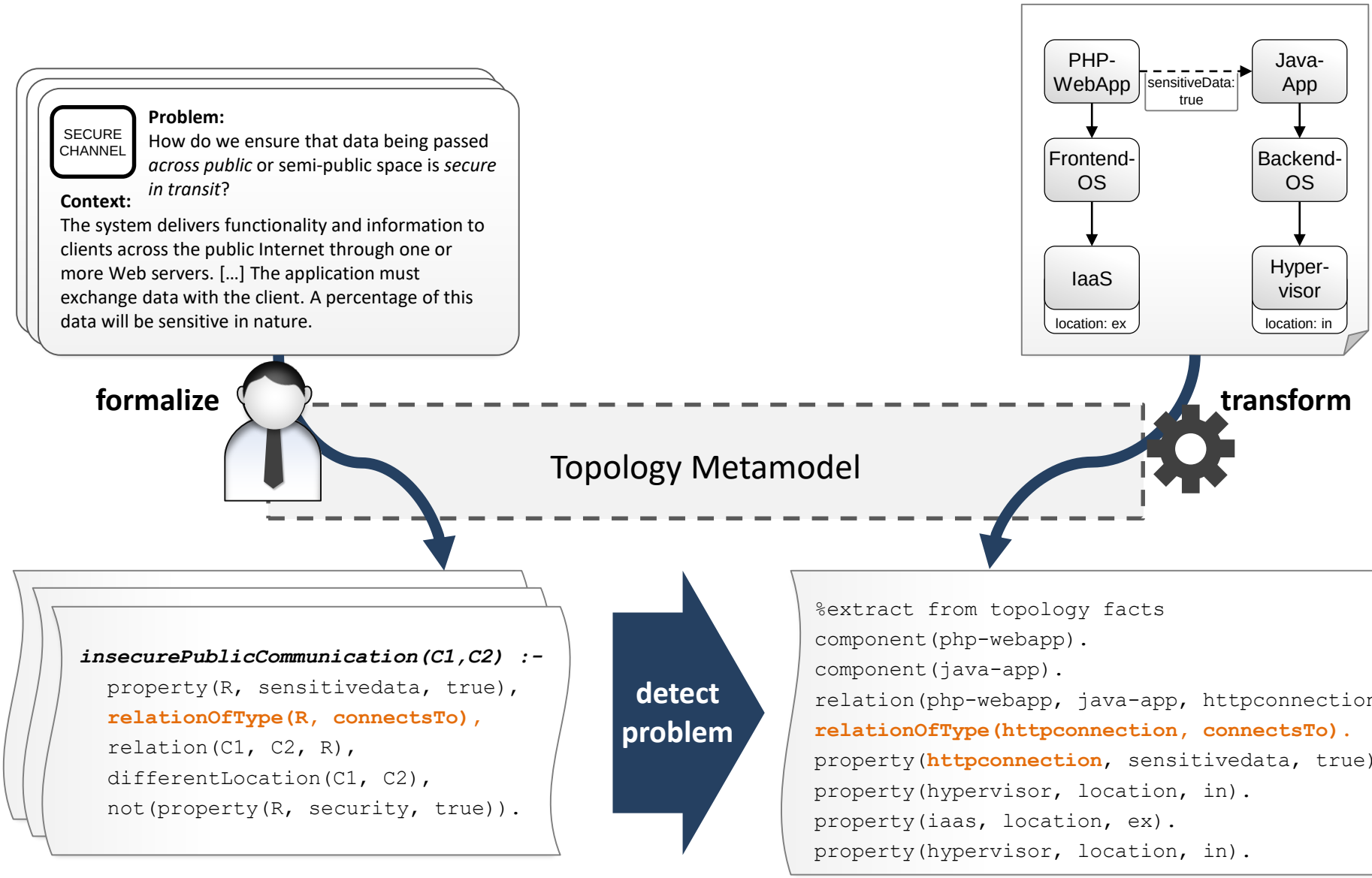
Problem Detection Approach



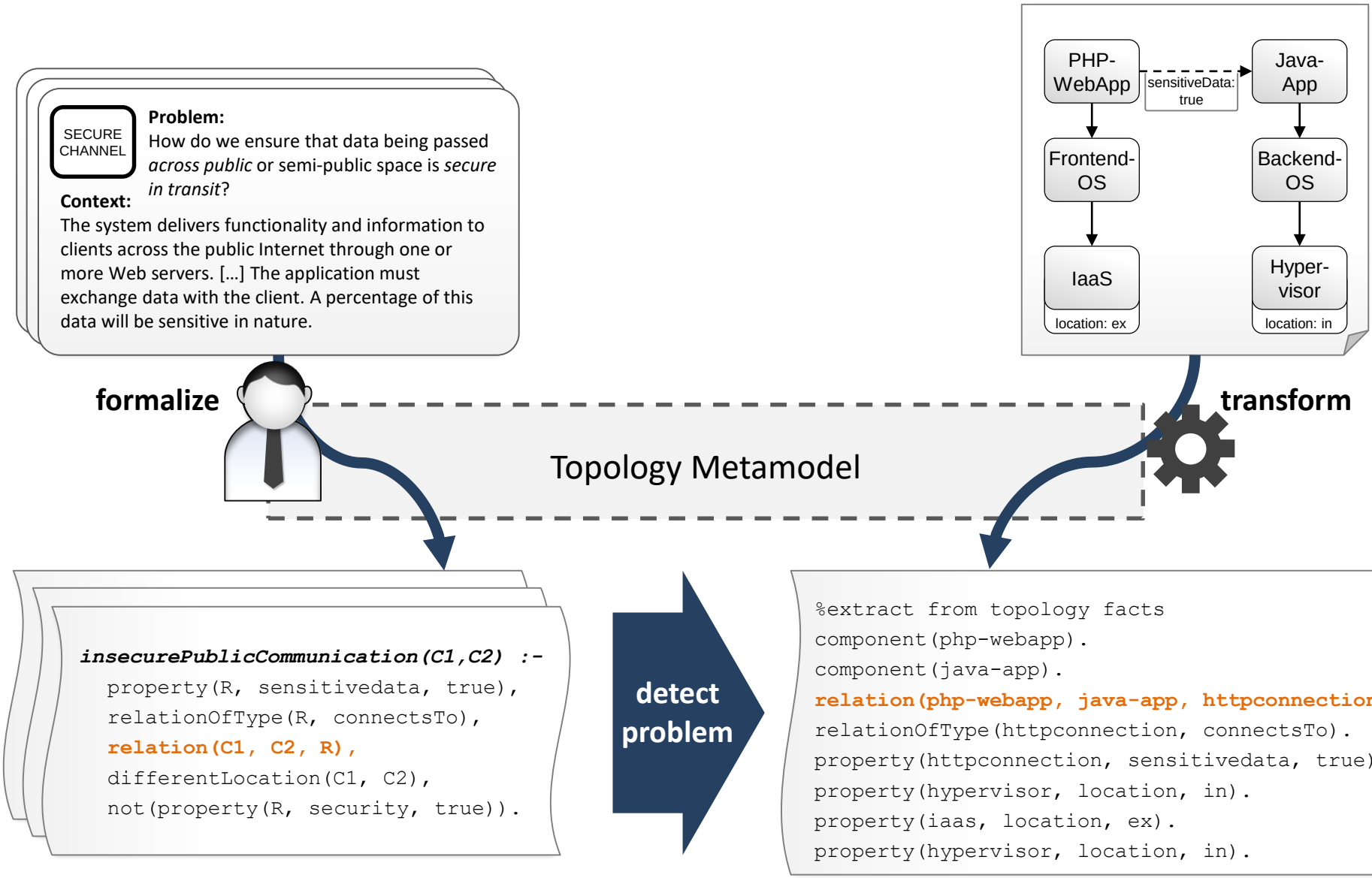
Problem Detection Approach



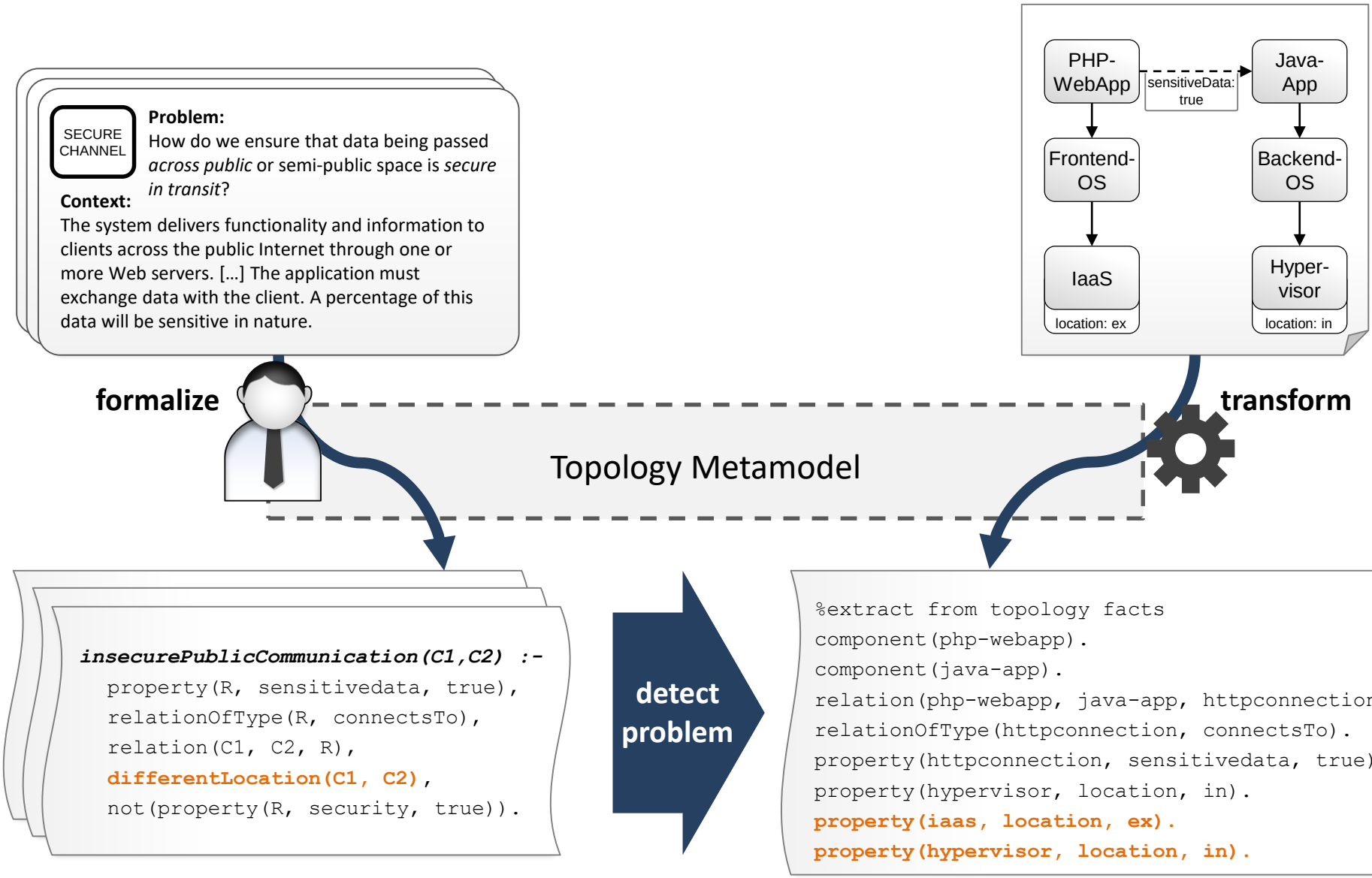
Problem Detection Approach



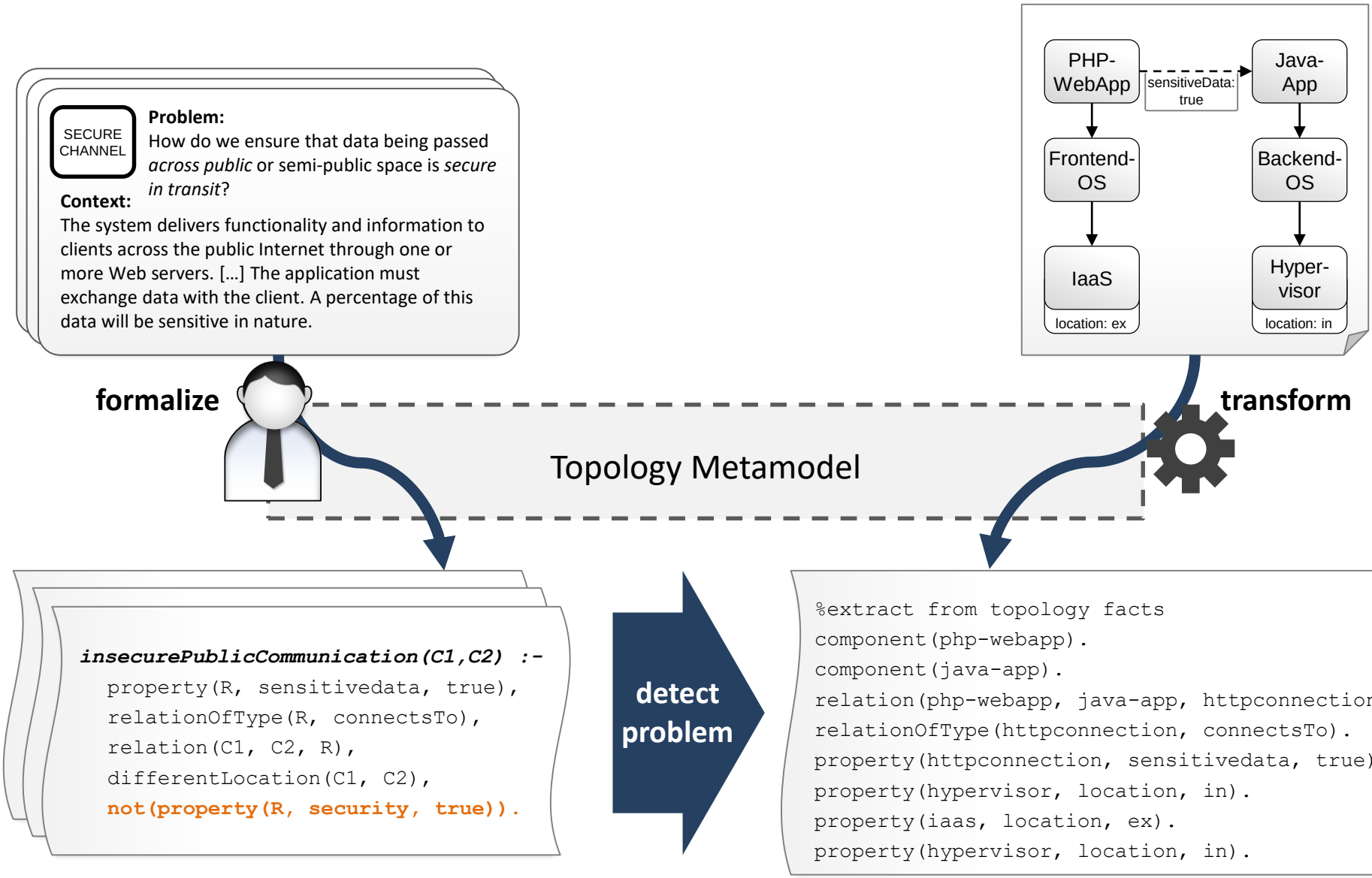
Problem Detection Approach



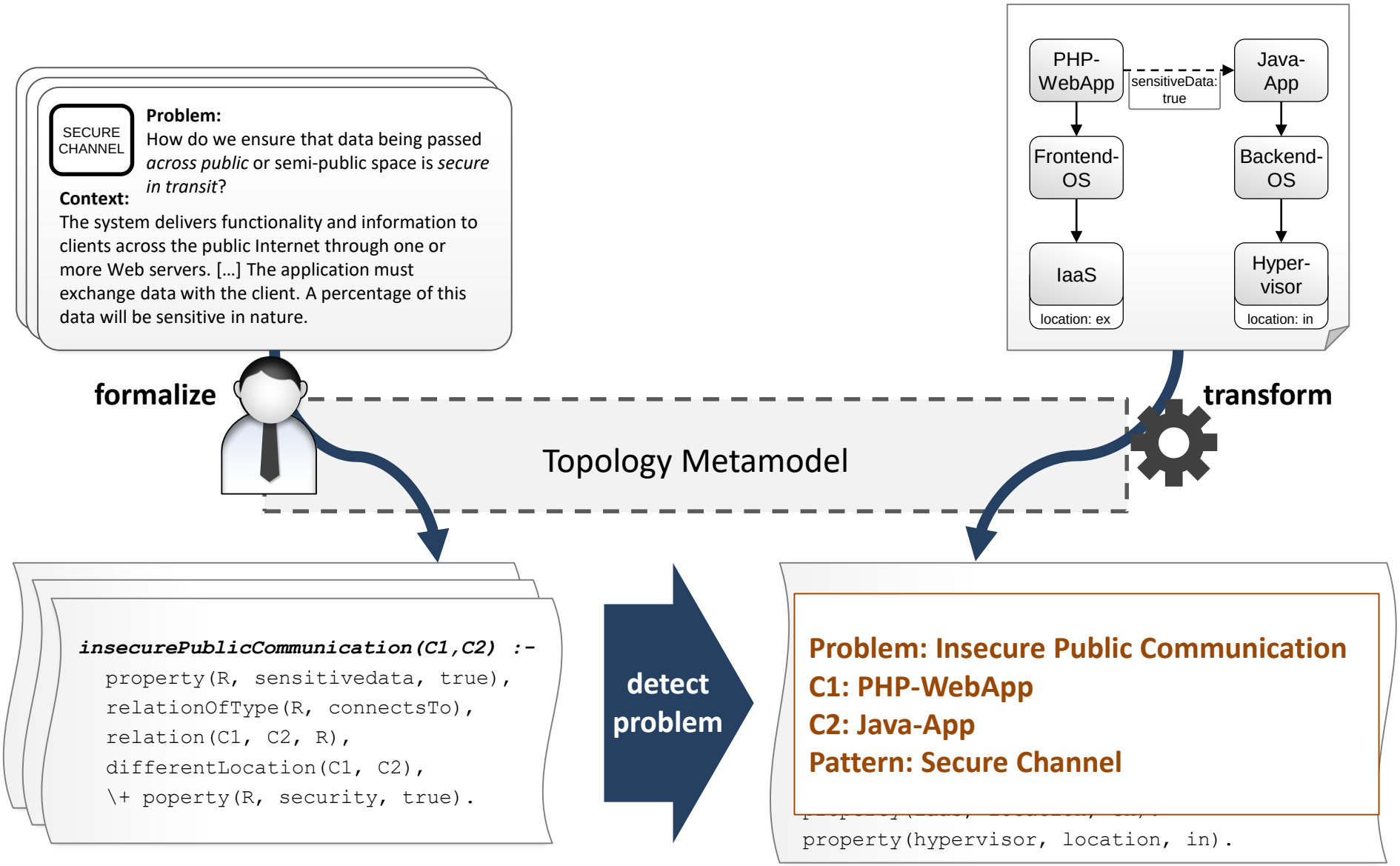
Problem Detection Approach



Problem Detection Approach



Problem Detection Approach



Prototypical Implementation Topology-ProDec*

- Based on TOSCA using the modeling tool Winery
- SWI Prolog

Secure Channel

Problem

Insecure Communication Channel

Problem Rule

```
insecure_public_communication(Component_1, Component_2) :-
        property(Relation_ID, sensitivedata, true),
        relation_of_type(Relation_ID, connectsto),
        relation(Component_1, Component_2, Relation_ID),
        components_in_different_locations(Component_1, Component_2),
        not(property(Relation_ID, security, true)).
```

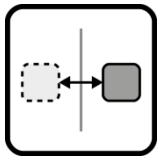
Solution Description

Create secure channels for sensitive data that obscure the data in transit. Exchange information between client and server to allow them to set up encrypted communication between themselves. [...]

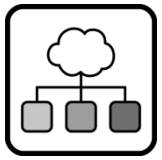
*https://github.com/saatkamp/topology-prodec

Application & Conclusion

- Prototypical validation based on 3 Cloud Computing and 1 Security Pattern



Application Component Proxy



Integration Provider

Message Mover

Secure Channel

- Future work:
  - Semantic model for the formalization and transformation process
  - Realization of solutions for the detected problems in topologies

Thank you for your attention and I hope I will see you at the poster session for interesting discussions ☺