UNIVERSITÄT KOBLENZ · LANDAU

Institute for Software Technology

Secure Data-Flow Compliance Checks between Models and Code based on Automated Mappings

UNIVERSITÄT KOBLENZ · LANDAU

UNIVERSITY OF GOTHENBURG

CHALMERS UNIVERSITY OF TECHNOLOGY

Fraunhofer

Sven Peldszus[1], Katja Tuma[2], Daniel Strüber[2], Jan Jürjens[1,3], Riccardo Scandariato[2]

1) University of Koblenz-Landau, Germany
2) University of Gothenburg and Chalmers University of Technology, Sweden
3) Fraunhofer Institute for Software and Systems Engineering, Germany

speldszus@uni-koblenz.de

Motivation

The New York Times

After a Data Breach, British Airways Faces a Record Fine

BBC Sign in News Sport Reel Worklife

NEWS

Home Video World UK Business Tech Science Stories

Technology

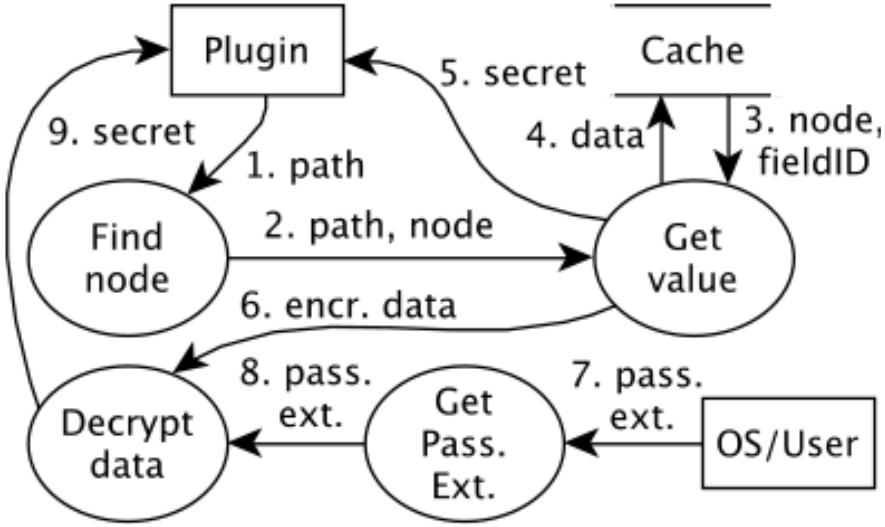
Apple's App Store infected with XcodeGhost malware in China

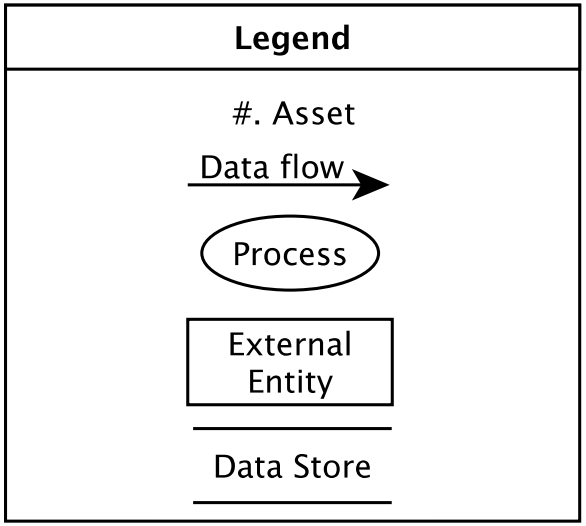
21 September 2015

- Security vulnerabilities can be expensive
- Nearly every kind of software can be affected

Model-based Security Engineering

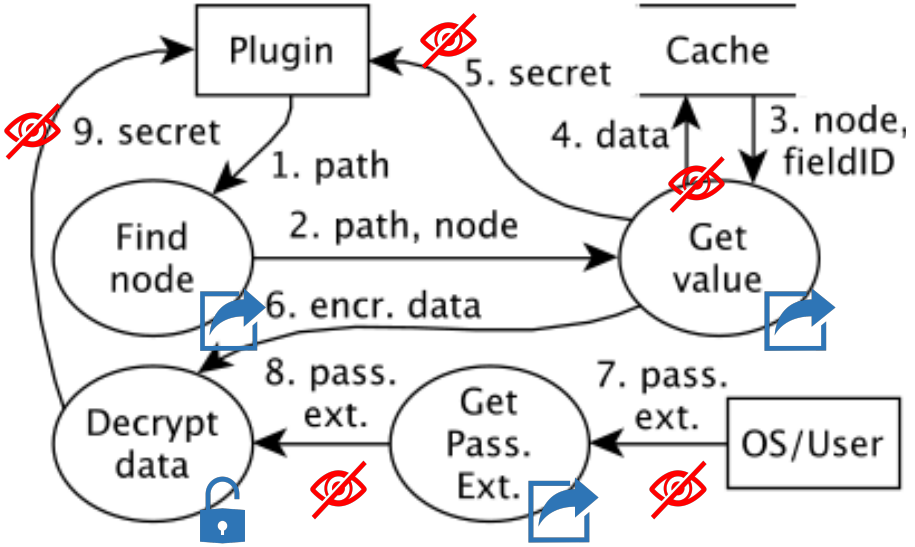
- **Security by design**: System's assets and threats have to be defined early
- Specification of the system (e.g. Eclipse Secure Storage) using a **Security Data Flow Diagram (SecDFD)**

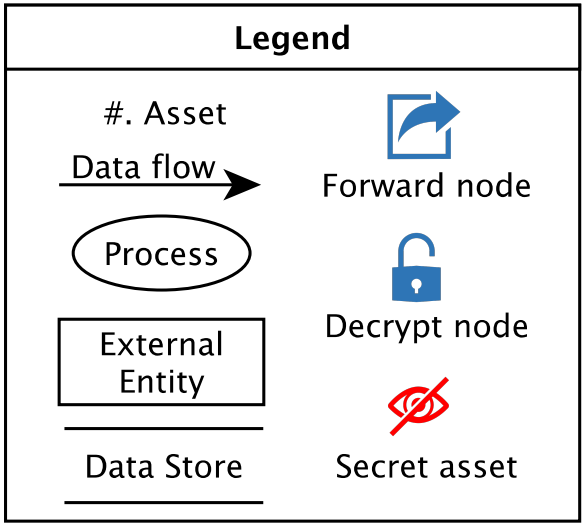




Model-based Security Engineering

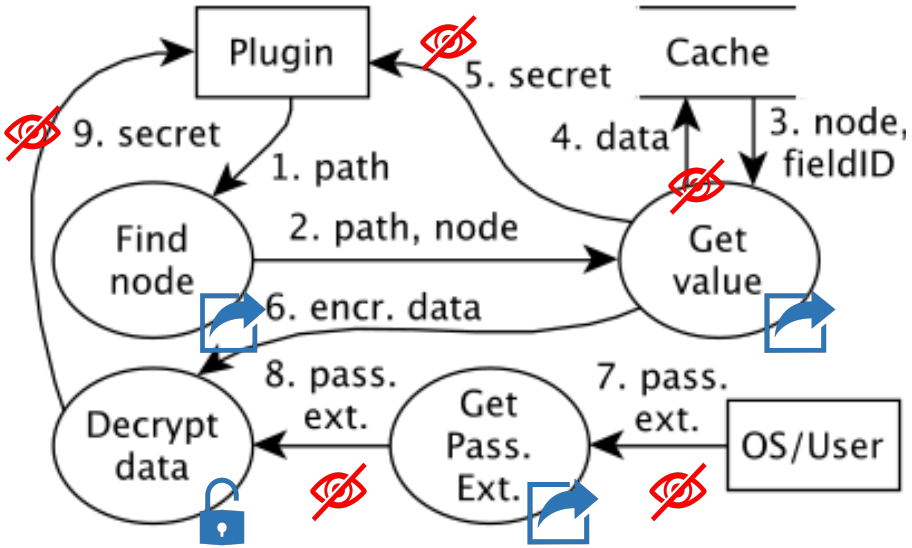
- **Security by design**: System's assets and threats have to be defined early
- Specification of the system (e.g. Eclipse Secure Storage) using a **Security Data Flow Diagram (SecDFD)**





Implementation of the Models

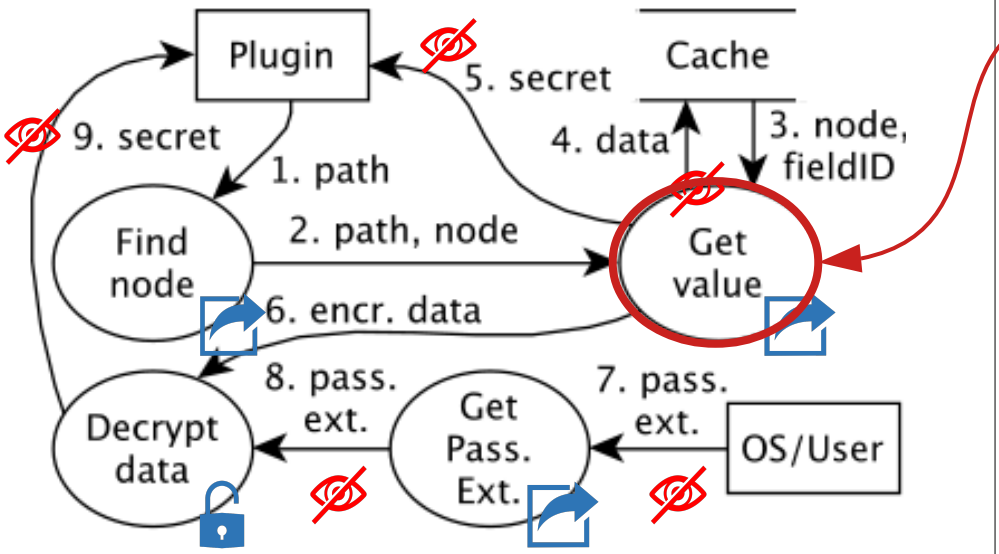
- **Security by design**: System's assets and threats have to be defined early
- Specification of the system (e.g. Eclipse Secure Storage) using a **Security Data Flow Diagram (SecDFD)**




```
 30  public class SecurePreferences {
       ...
251    public String get(String key, String def,
       SecurePreferencesContainer container) ... {
         ...
255      String encryptedValue = internalGet(key);
256      if (encryptedValue == null)
257        return null;
258
259      CryptoData data =
           new CryptoData(encryptedValue);
260      String moduleID = data.getModuleID();
         ...
267      PasswordExt passwordExt =
           getRoot().getPassword(moduleID,
            container, false);
         ...
277    }
       ...
508  }
```

Compliance between Models & Code

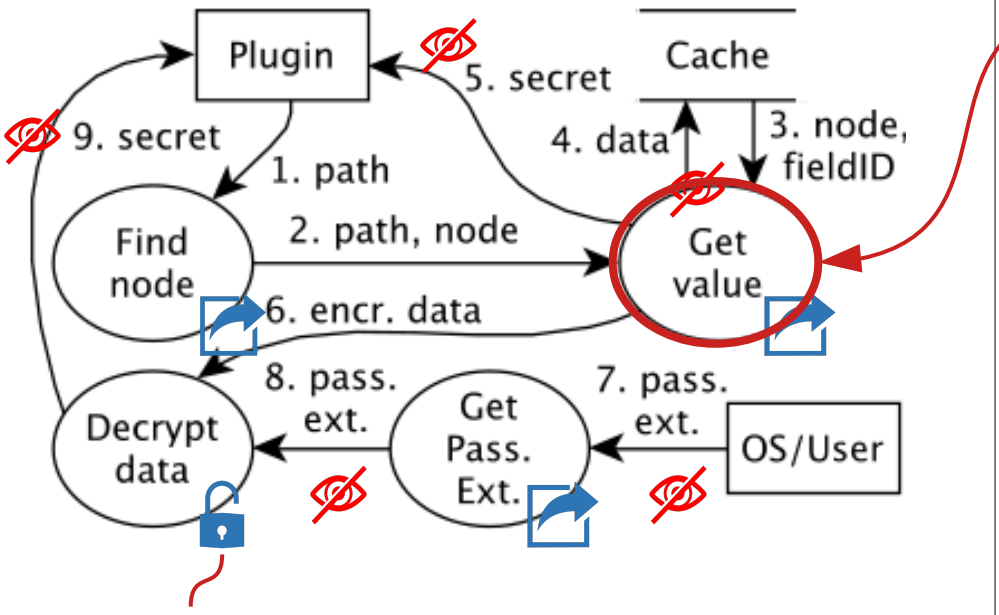
- **Security by design**: System's assets and threats have to be defined early
- Specification of the system (e.g. Eclipse Secure Storage) using a **Security Data Flow Diagram (SecDFD)**




```
 30  public class SecurePreferences {
       ...
251    public String get(String key, String def,
       SecurePreferencesContainer container) ... {
         ...
255      String encryptedValue = internalGet(key);
256      if (encryptedValue == null)
257        return null;
258
259      CryptoData data =
           new CryptoData(encryptedValue);
260      String moduleID = data.getModuleID();
         ...
267      PasswordExt passwordExt =
           getRoot().getPassword(moduleID,
            container, false);
         ...
277    }
       ...
508  }
```

Compliance between Models & Code

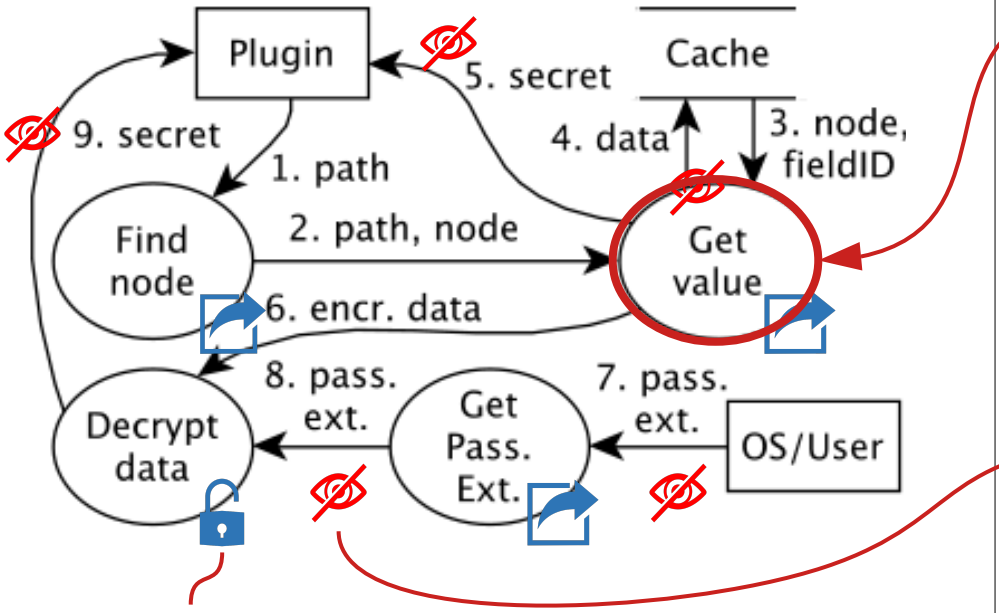
- **Security by design**: System's assets and threats have to be defined early
- Specification of the system (e.g. Eclipse Secure Storage) using a **Security Data Flow Diagram (SecDFD)**




```
 30  public class SecurePreferences {
       ...
251    public String get(String key, String def,
       SecurePreferencesContainer container) ... {
         ...
255      String encryptedValue = internalGet(key);
256      if (encryptedValue == null)
257        return null;
258
259      CryptoData data =
           new CryptoData(encryptedValue);
260      String moduleID = data.getModuleID();
         ...
267      PasswordExt passwordExt =
           getRoot().getPassword(moduleID,
           container, false);
         ...
277    }
       ...
508  }
```

Compliance between Models & Code

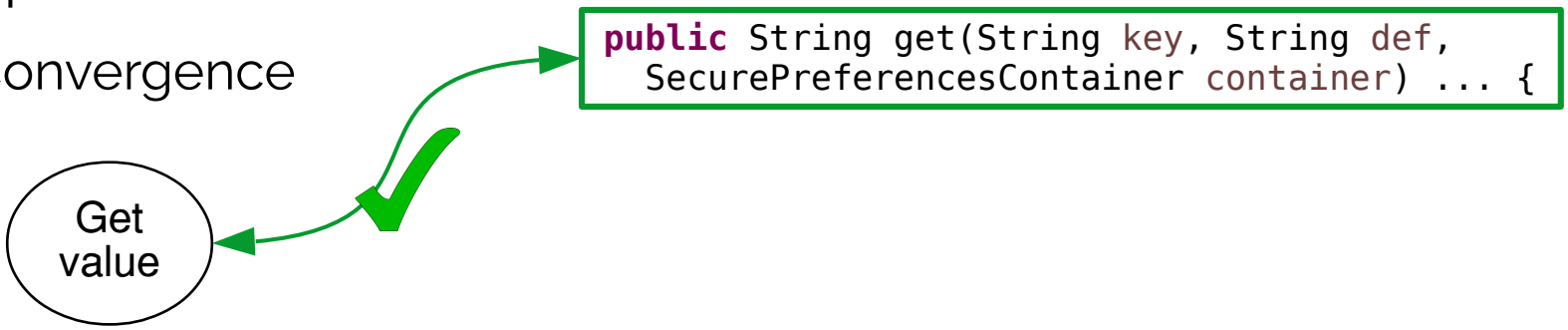
- **Security by design**: System's assets and threats have to be defined early
- Specification of the system (e.g. Eclipse Secure Storage) using a **Security Data Flow Diagram (SecDFD)**




```
 30  public class SecurePreferences {
       ...
251    public String get(String key, String def,
       SecurePreferencesContainer container) ... {
         ...
255      String encryptedValue = internalGet(key);
256      if (encryptedValue == null)
257        return null;
258
259      CryptoData data =
           new CryptoData(encryptedValue);
260      String moduleID = data.getModuleID();
         ...
267      PasswordExt passwordExt =
           getRoot().getPassword(moduleID,
           container, false);
         ...
277    }
       ...
508  }
```

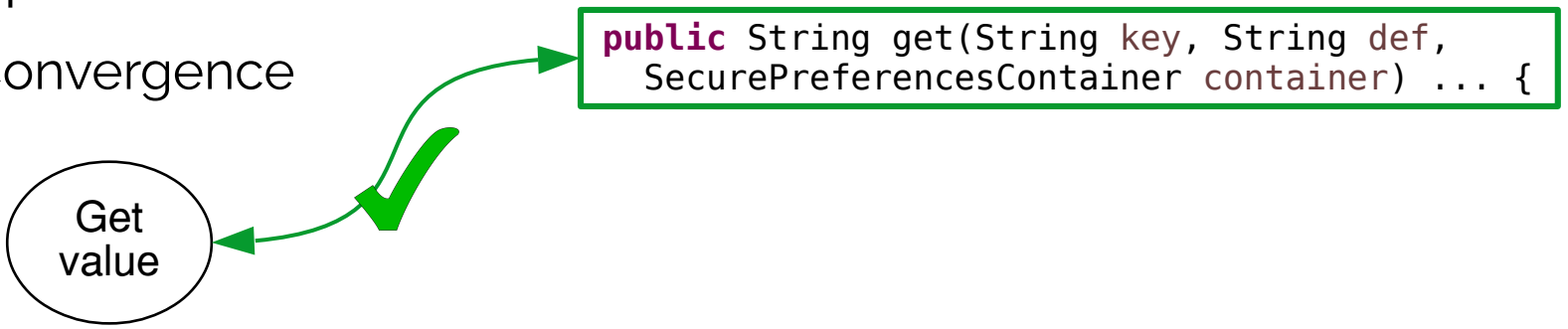
Compliance between Models & Code

- Compliance Checks
  - Convergence

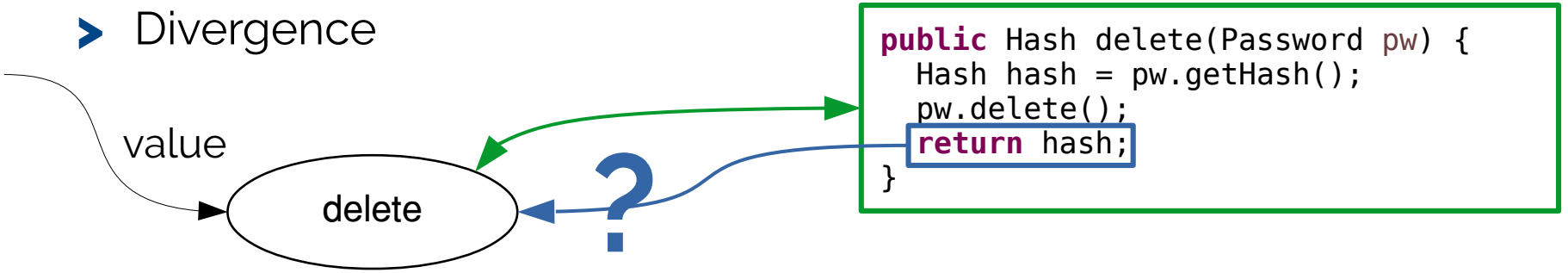


Compliance between Models & Code

- Compliance Checks
  - Convergence

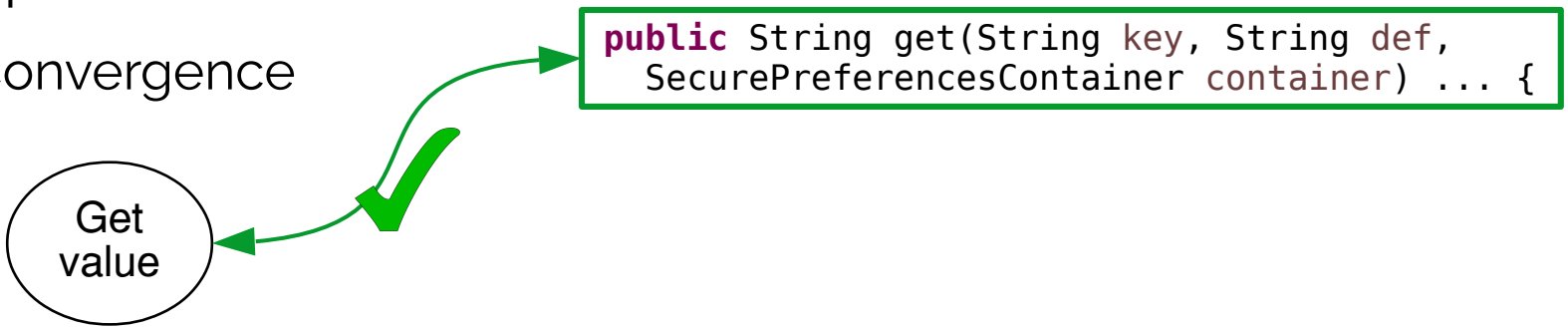


  - Divergence

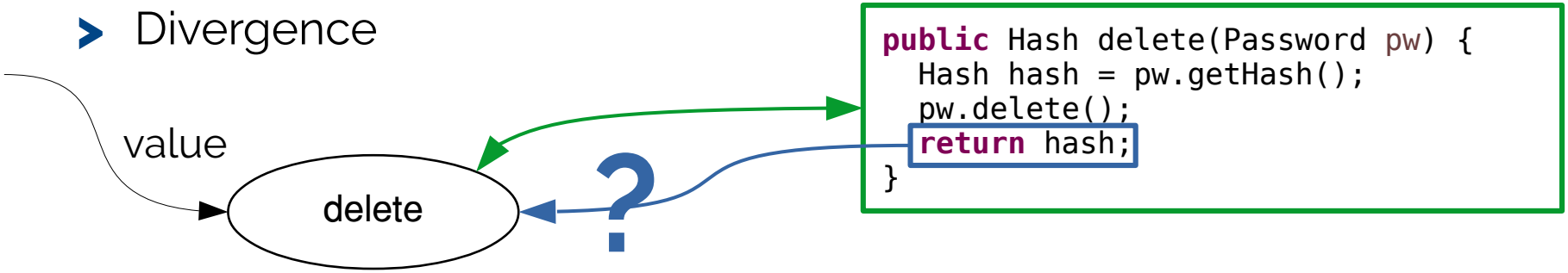


Compliance between Models & Code

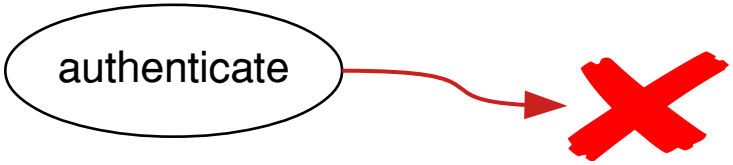
- Compliance Checks
  - Convergence



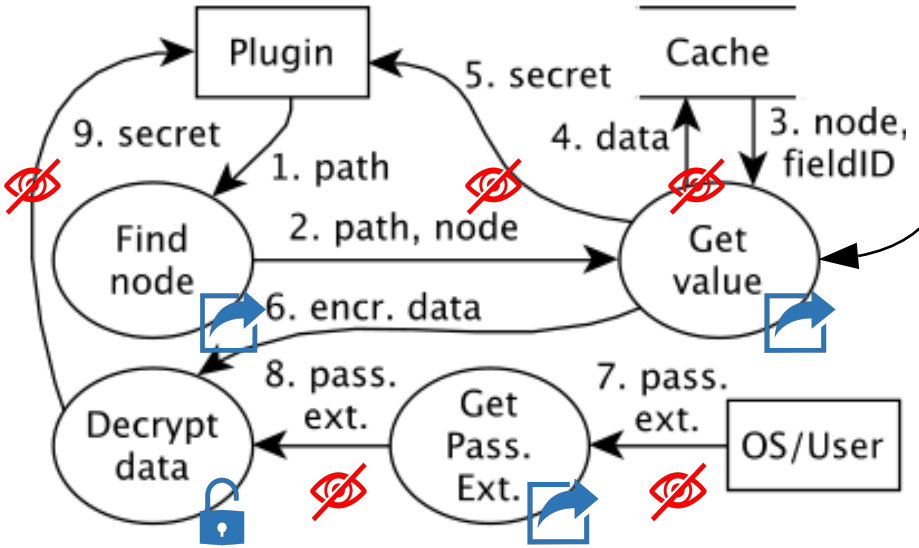
  - Divergence



  - Absence



Security Analysis of Source Code



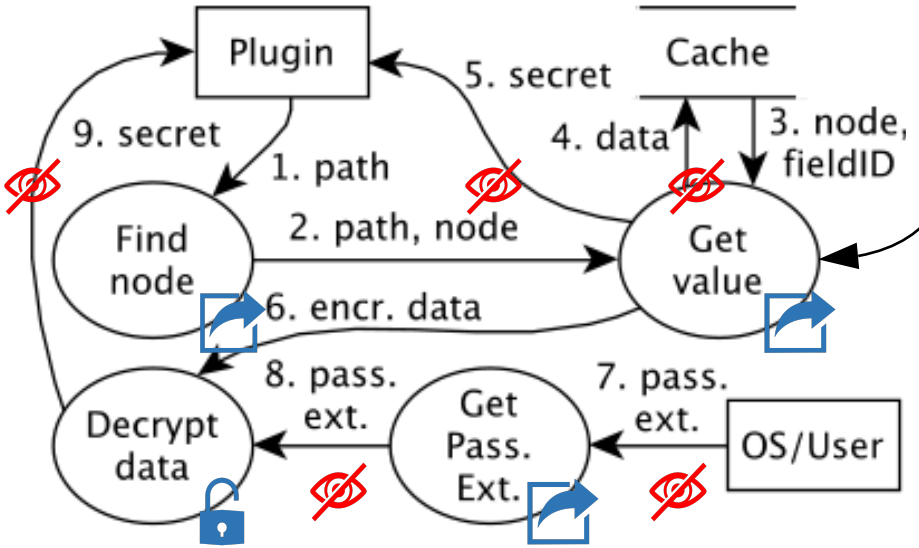
OWASP

A3-Sensitive Data Exposure

```
 30  public class SecurePreferences {
       ...
251    public String get(String key, String def,
       SecurePreferencesContainer container) ... {
         ...
255      String encryptedValue = internalGet(key);
256      if (encryptedValue == null)
257        return null;
258
259      CryptoData data =
           new CryptoData(encryptedValue);
260      String moduleID = data.getModuleID();
         ...
267      PasswordExt passwordExt =
           getRoot().getPassword(moduleID,
            container, false);
         ...
277    }
       ...
508  }
```

Security Analysis of Source Code

- Based on Security Labels
  - Calculate Security Metrics – e.g. Number of Classes containing Secret Data
  - Secure Data Flow Analysis
  - Runtime-Monitoring

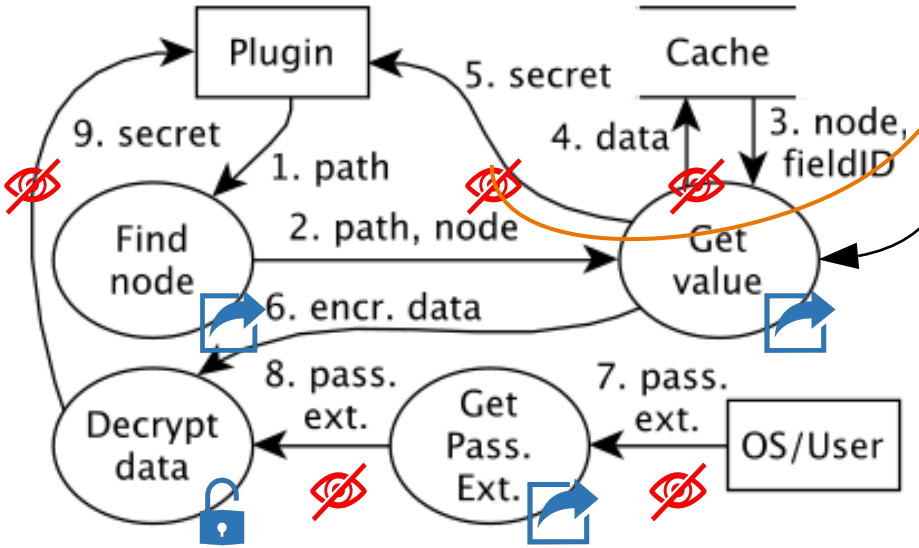


OWASP
A3-Sensitive Data Exposure

```
 30  public class SecurePreferences {
       ...
251    public String get(String key, String def,
       SecurePreferencesContainer container) ... {
         ...
255      String encryptedValue = internalGet(key);
256      if (encryptedValue == null)
257        return null;
258
259      CryptoData data =
           new CryptoData(encryptedValue);
260      String moduleID = data.getModuleID();
         ...
267      PasswordExt passwordExt =
           getRoot().getPassword(moduleID,
            container, false);
         ...
277    }
       ...
508  }
```

Security Analysis of Source Code

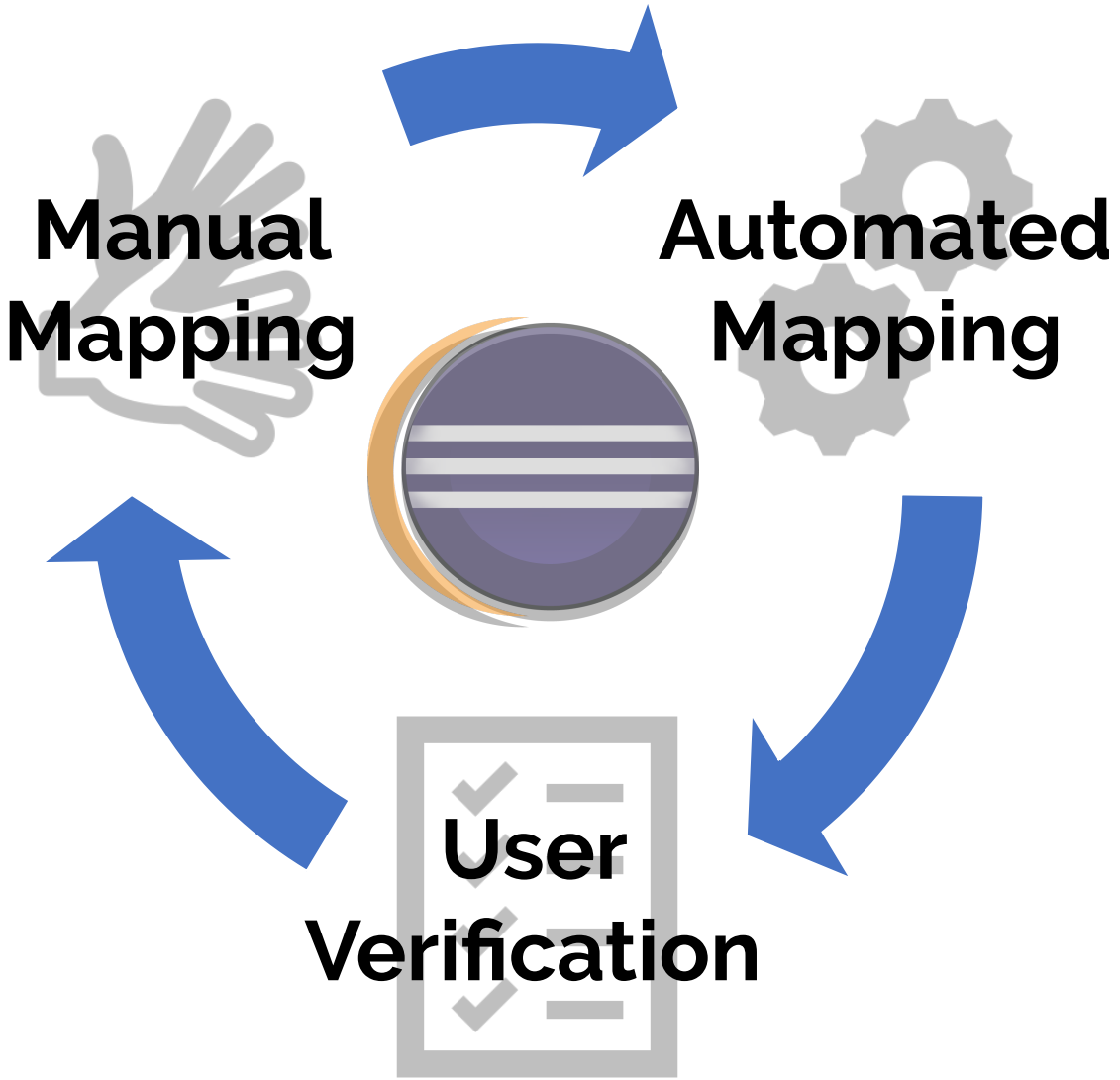
- Based on Security Labels
  - Calculate Security Metrics – e.g. Number of Classes containing Secret Data
  - Secure Data Flow Analysis
  - Runtime-Monitoring



OWASP
A3-Sensitive Data Exposure

```
 30  public class SecurePreferences {
       @Secrecy
251    public String get(String key, String def,
       SecurePreferencesContainer container) ... {
         ...
255      String encryptedValue = internalGet(key);
256      if (encryptedValue == null)
257        return null;
258
259      CryptoData data =
           new CryptoData(encryptedValue);
260      String moduleID = data.getModuleID();
         ...
267      PasswordExt passwordExt =
           getRoot().getPassword(moduleID,
            container, false);
         ...
277    }
       ...
508  }
```

Mapping Generation: Semi-Automated Process

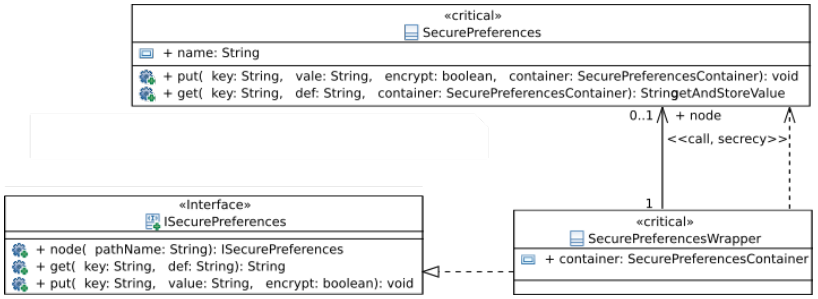


Abstract Program Representation for Design Flaw Detection: Program Model

UNIVERSITÄT KOBLENZ · LANDAU

Institute for Software Technology

UML Model

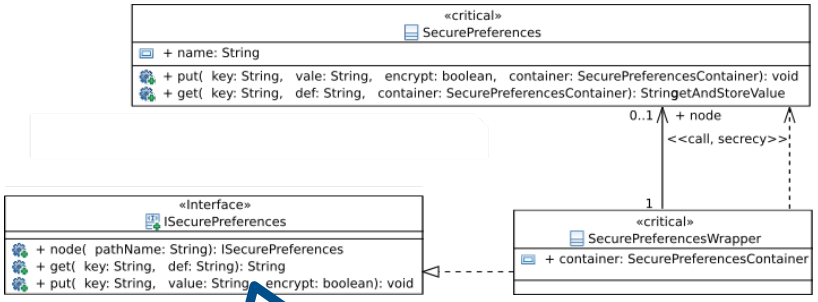


Abstract Program Representation for Design Flaw Detection: Program Model

UNIVERSITÄT KOBLENZ · LANDAU

ist Institute for Software Technology

UML Model



Misses crucial semantic information:

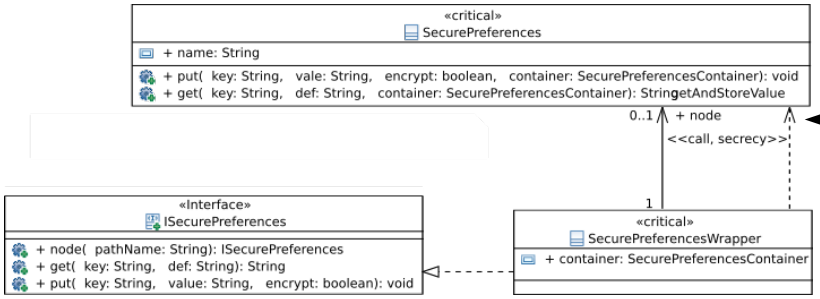
- method calls
- field accesses
- ...

Abstract Program Representation for Design Flaw Detection: Program Model

UNIVERSITÄT KOBLENZ · LANDAU

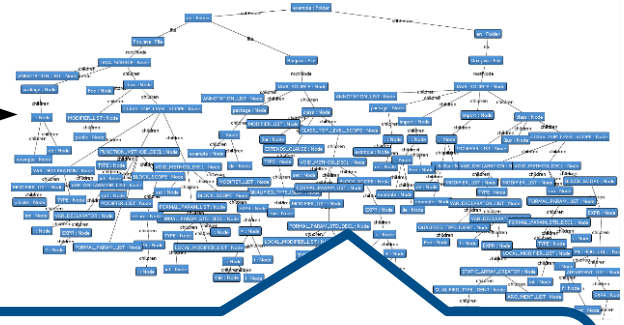
ist Institute for Software Technology

UML Model



abstraction

Abstract Syntax Tree



Too detailed and impracticable:

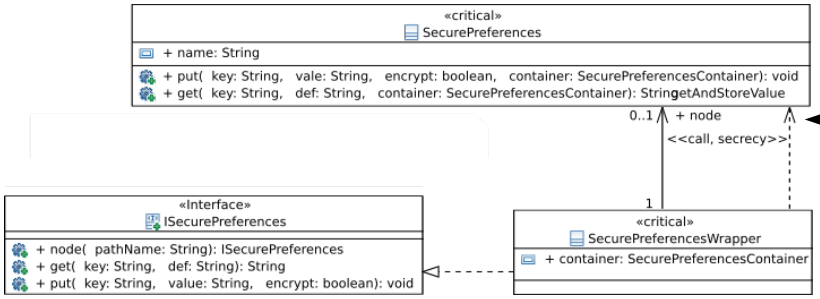
- complete method implementations
- no simple entry points

Abstract Program Representation for Design Flaw Detection: Program Model

UNIVERSITÄT KOBLENZ · LANDAU

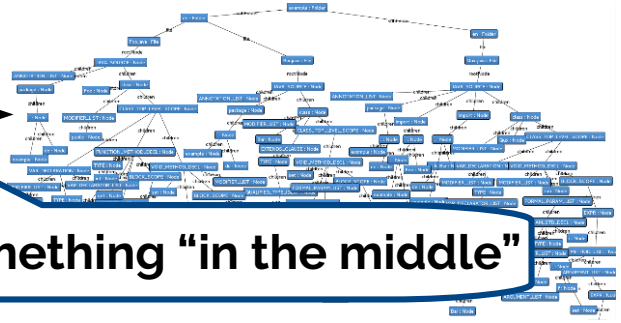
ist Institute for Software Technology

UML Model



abstraction

Abstract Syntax Tree



We need something “in the middle”

Abstract Program Representation for rogram Model

UNIVERSITÄT KOBLENZ · LANDAU

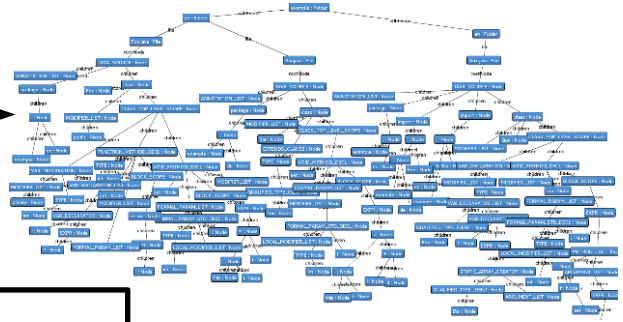
ist Institute for Software Technology

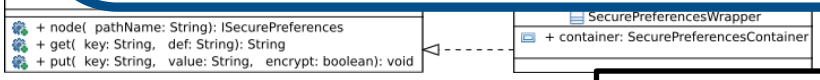
GRaViTY program model:

- **abstraction** with focus on class/method/field level
- enriched with **semantic information**:
  - overloading/overriding
  - method/field access

traction

Abstract Syntax Tree





Abstract Program Representation for rogram Model

UNIVERSITÄT KOBLENZ · LANDAU

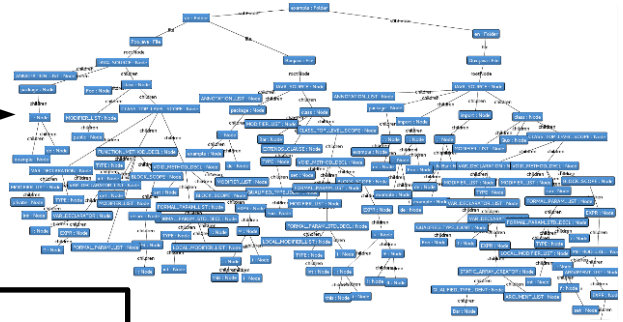
ist Institute for Software Technology

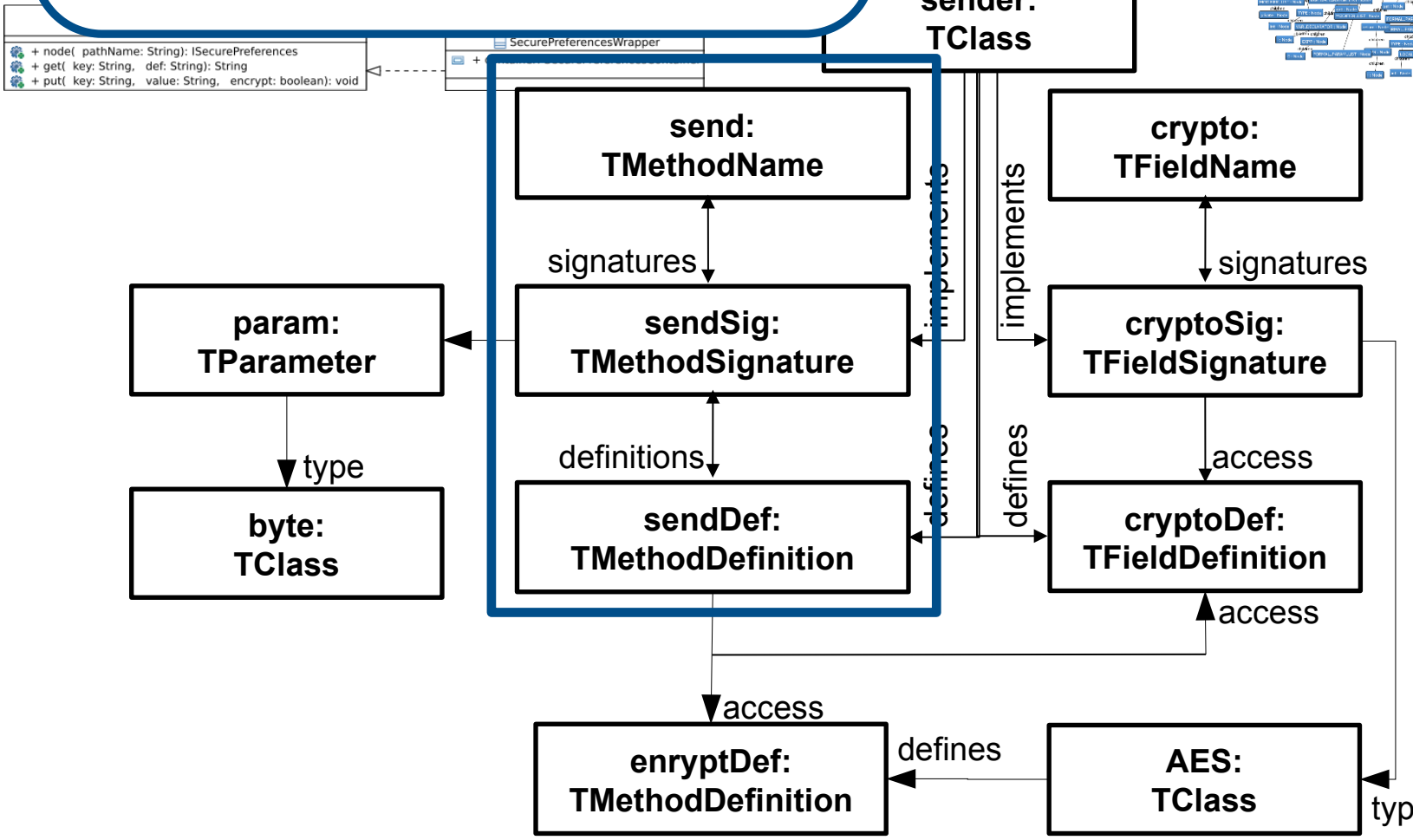
GRaViTY program model:

- **abstraction** with focus on class/method/field level
- enriched with **semantic information**:
  - overloading/overriding
  - method/field access

traction

Abstract Syntax Tree





Abstract Program Representation for rogram Model

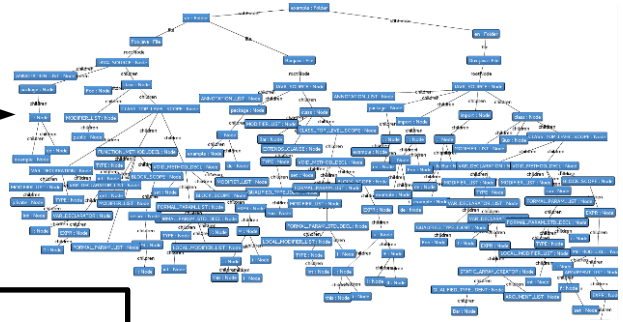
UNIVERSITÄT KOBLENZ · LANDAU

ist Institute for Software Technology

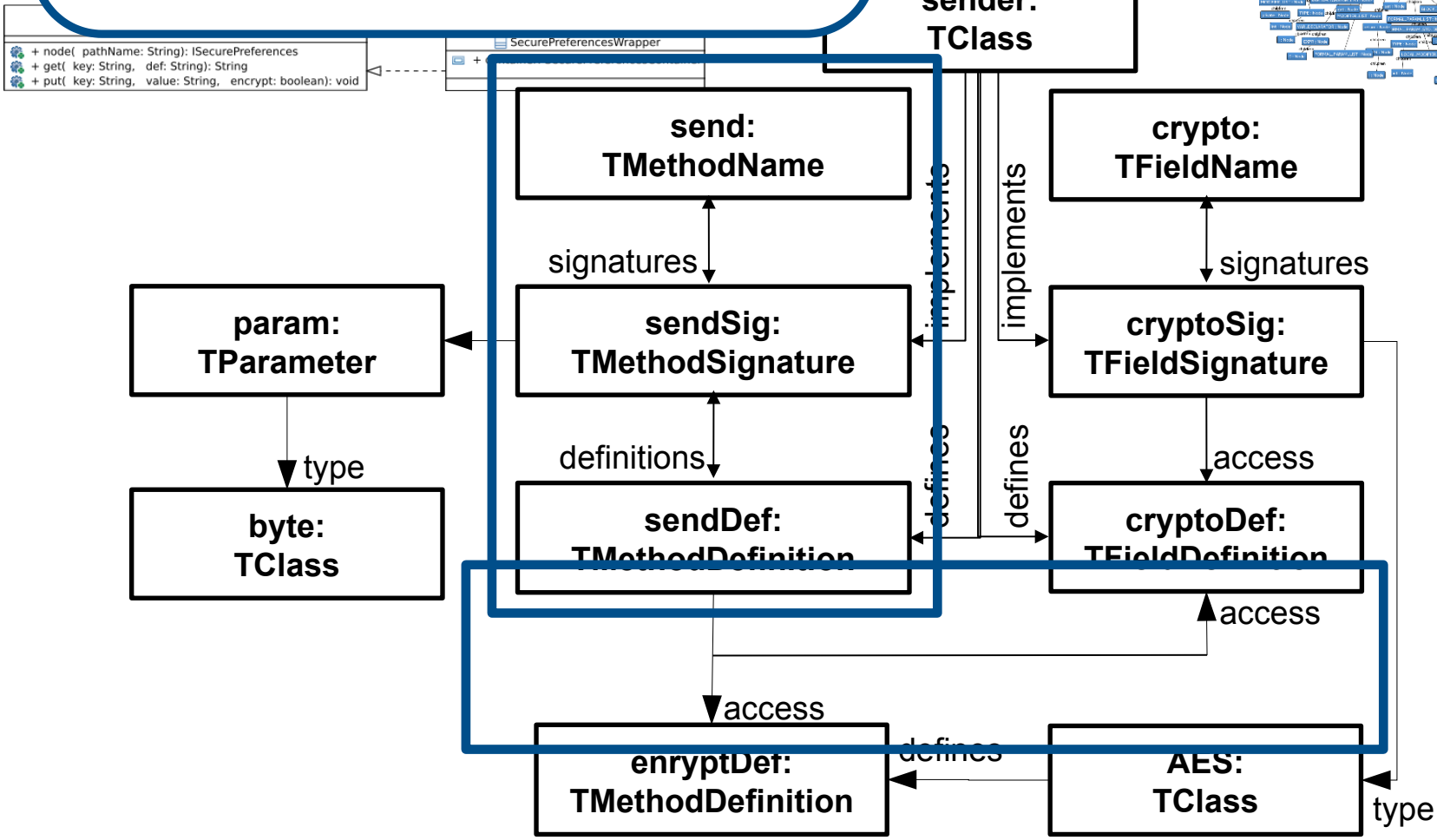
GRaViTY program model:

- **abstraction** with focus on class/method/field level
- enriched with **semantic information**:
  - overloading/overriding
  - method/field access

Abstract Syntax Tree



traction



Mapping a DFD to the Program Model

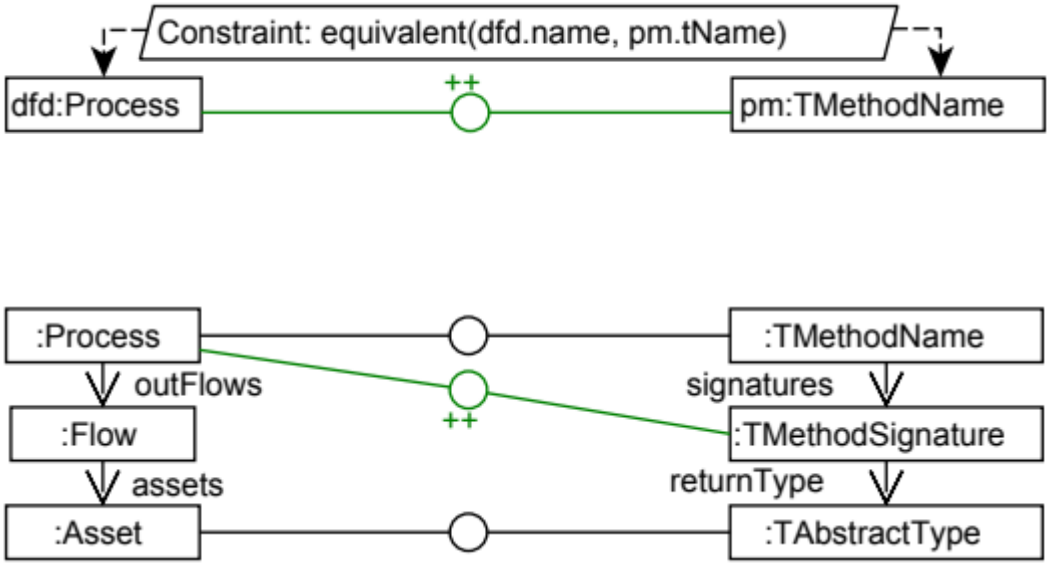
- Iterative mapping

Mapping a DFD to the Program Model

- Iterative mapping
  - 1) Mapping of Names:
    - Using a Heuristic
    - Selecting all with a certainty higher than 70%

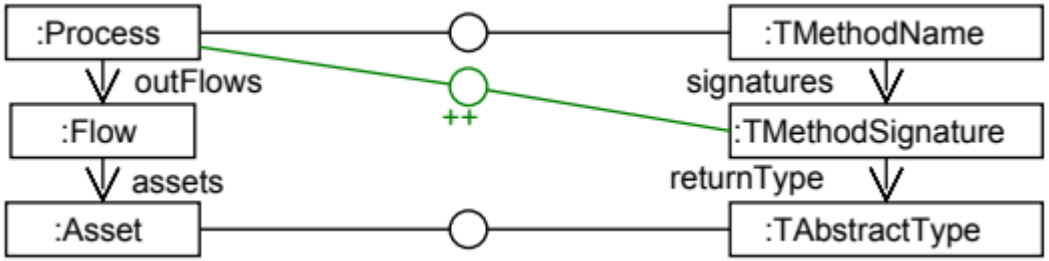
Mapping a DFD to the Program Model

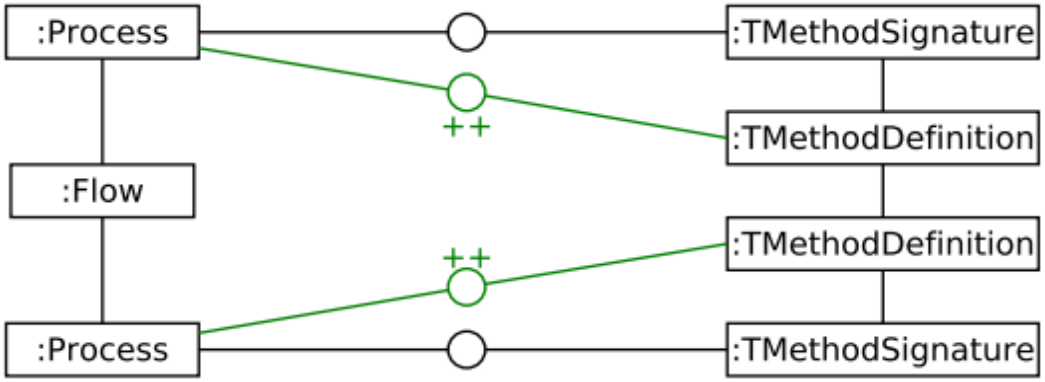
- Iterative mapping
  - 1) Mapping of Names:
    - Using a Heuristic
    - Selecting all with a certainty higher than 70%
  - 2) Mapping of Signatures:
    - Selecting all with at least one parameter or return type mapping
    - Certainty based on number of matches and name matching



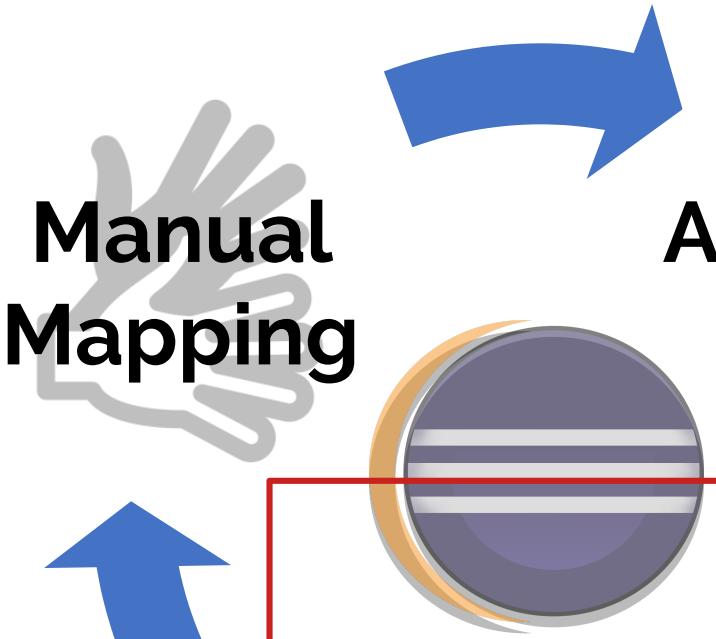
Mapping a DFD to the Program Model

- Iterative mapping
  - 1) Mapping of Names:
    - Using a Heuristic
    - Selecting all with a certainty higher than 70%
  - 2) Mapping of Signatures:
    - Selecting all with at least one parameter or return type mapping
    - Certainty based on number of matches and name matching
  - 3) Mapping of Definitions:
    - Certainty based on signature mappings





Mapping Generation: Semi-Automated Process

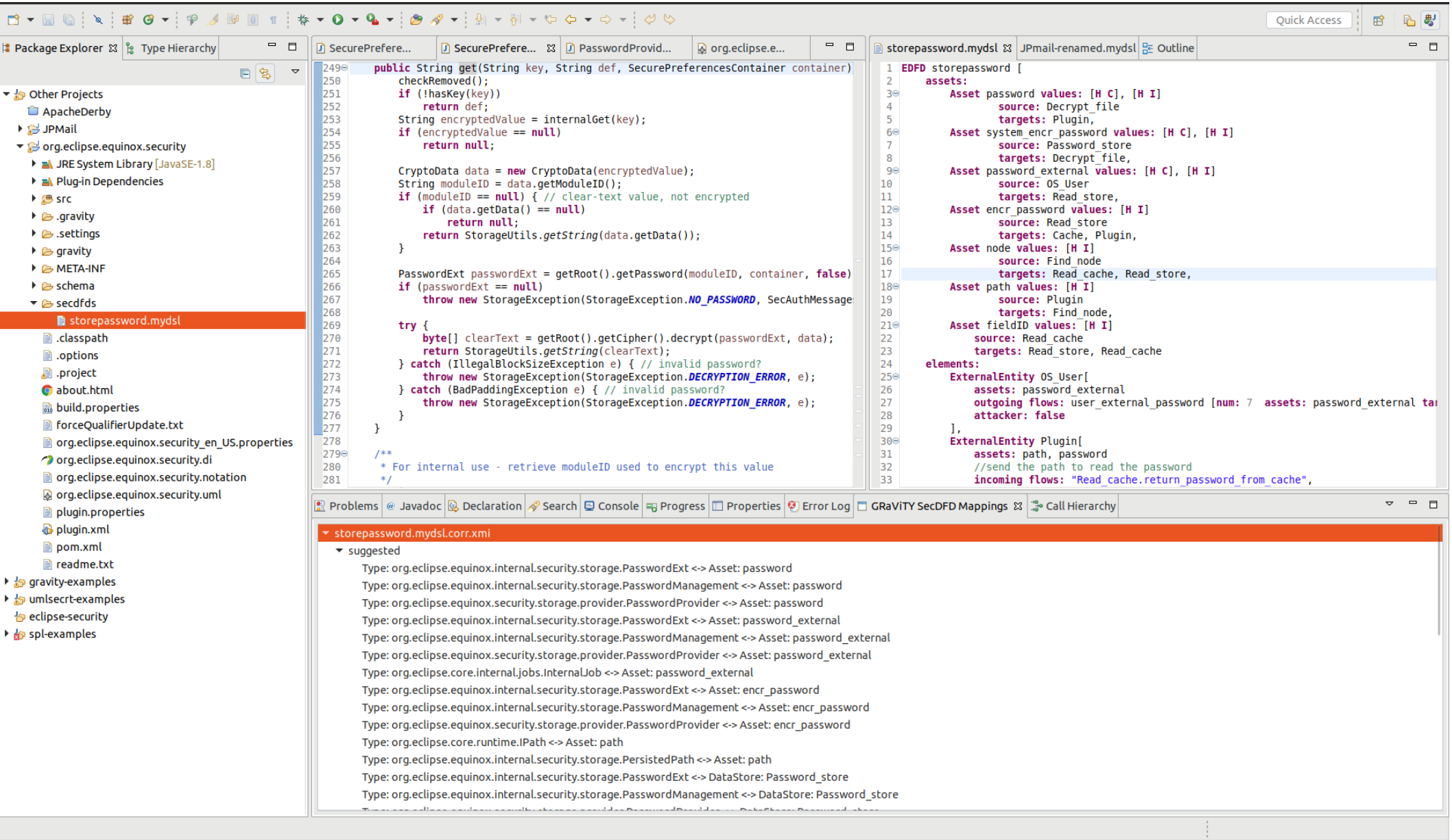


Present mappings with certainty >= 70%

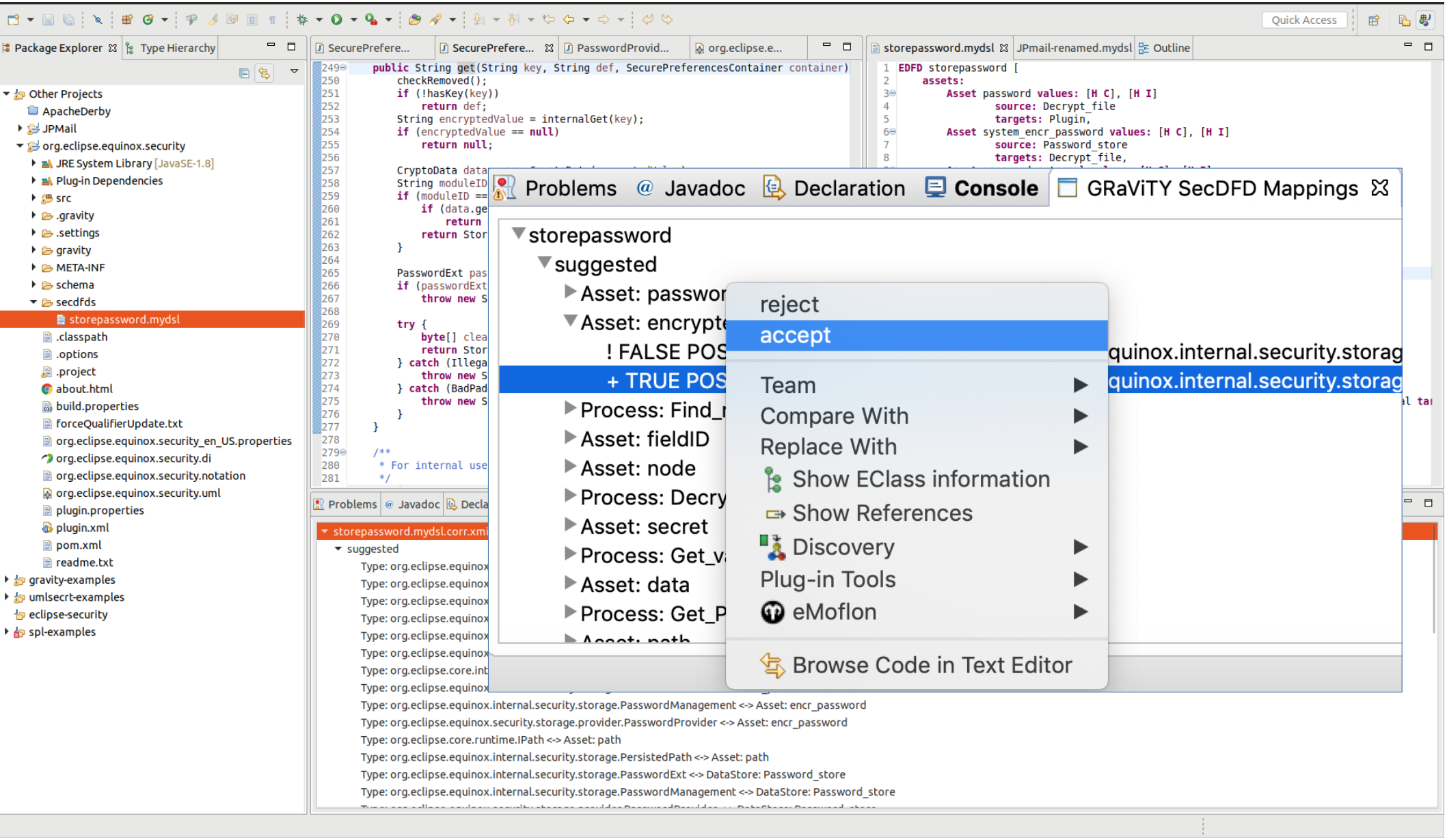
But only:

- Process → Definiton
- Asset → Type

User Verification

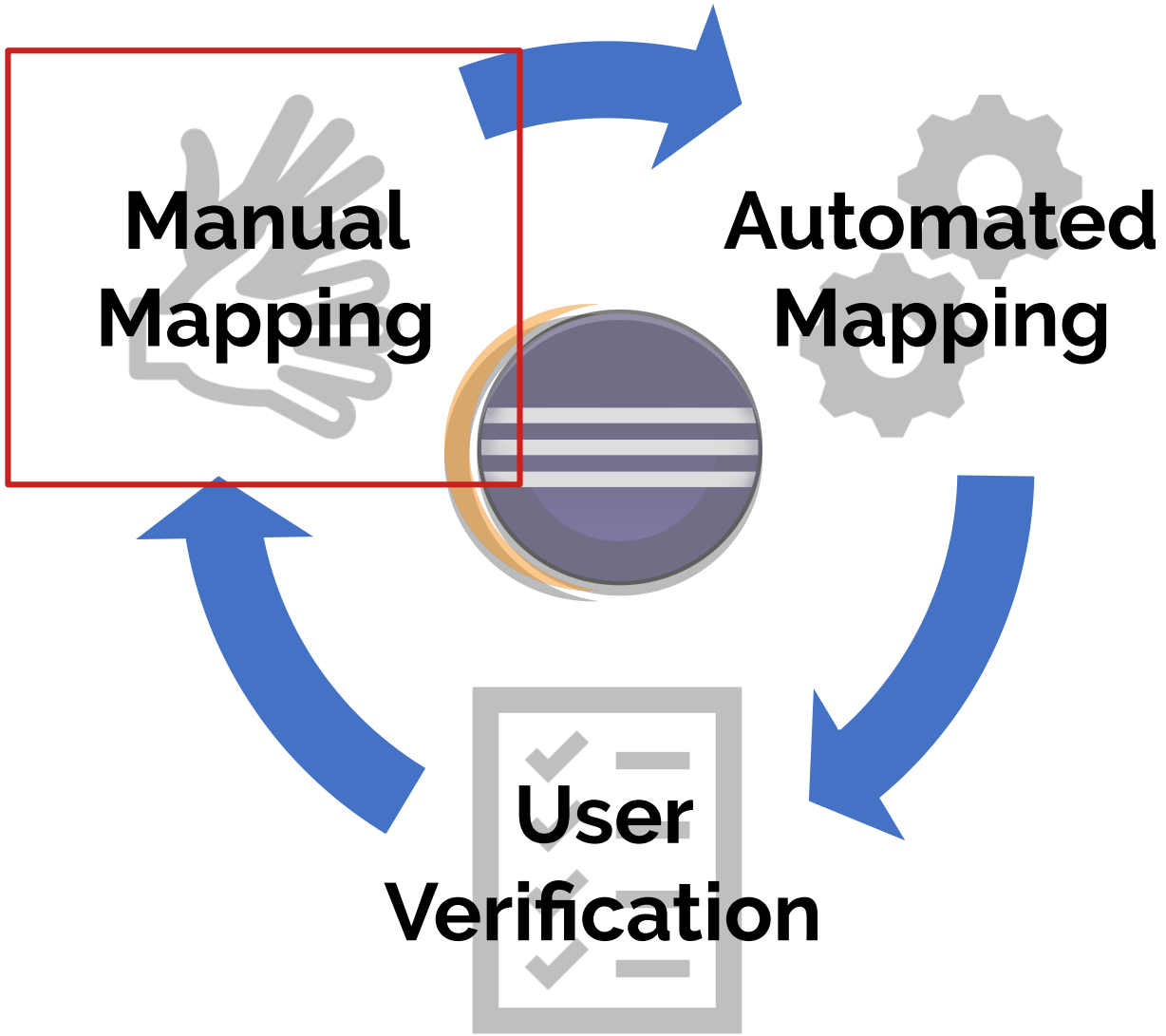


User Verification

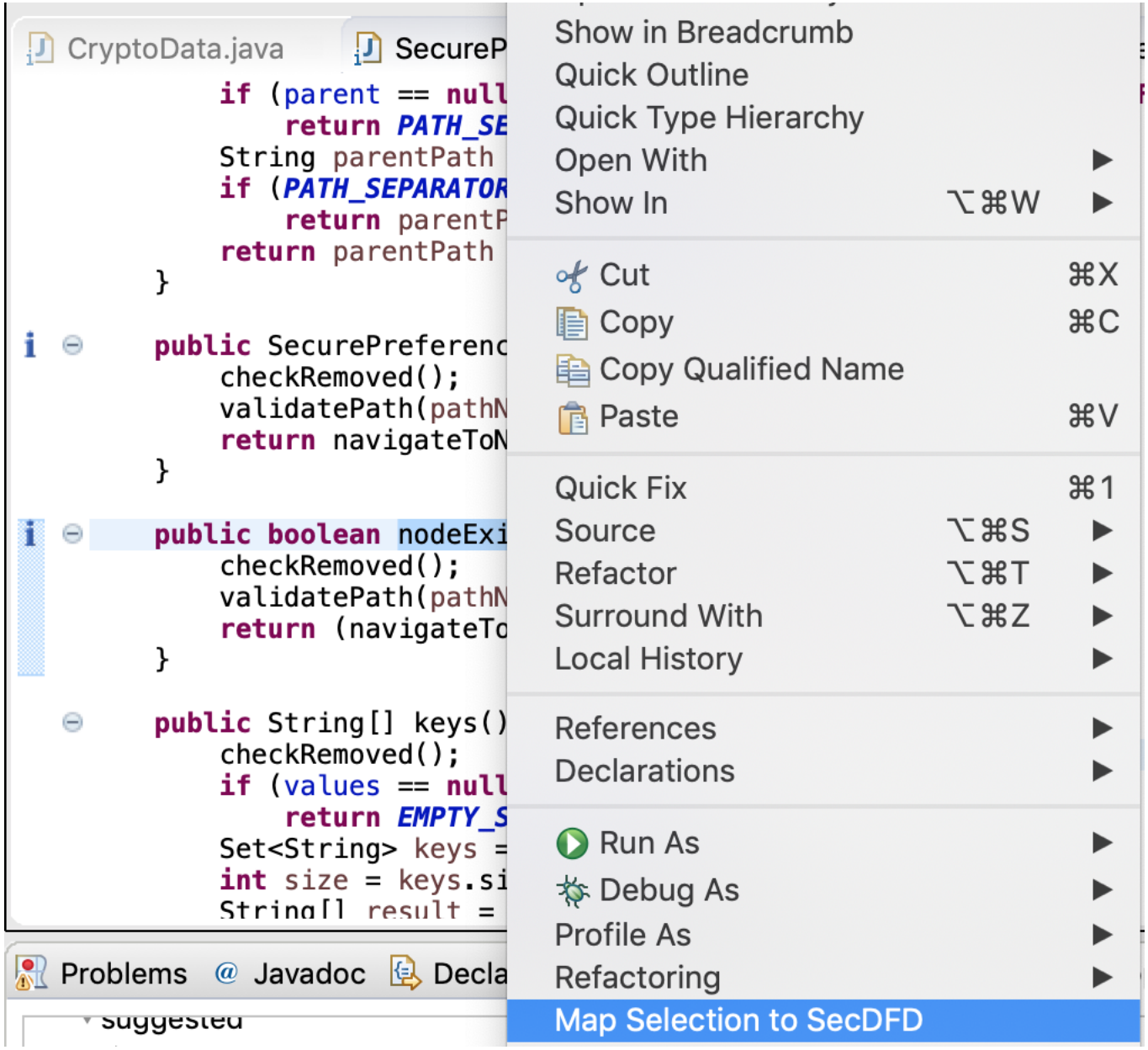


Mapping Generation: Semi-Automated Process

UNIVERSITÄT KOBLENZ · LANDAU
ist Institute for Software Technology



Manual Mapping



Evaluation

- **RQ1**. What is the correctness of the automated mappings?
  - In terms of precision and recall

- **RQ2**. What is the impact of the user on the correctness of mappings?
  - How much is the recall increased in each iteration by
    - the user
    - the automated mappings

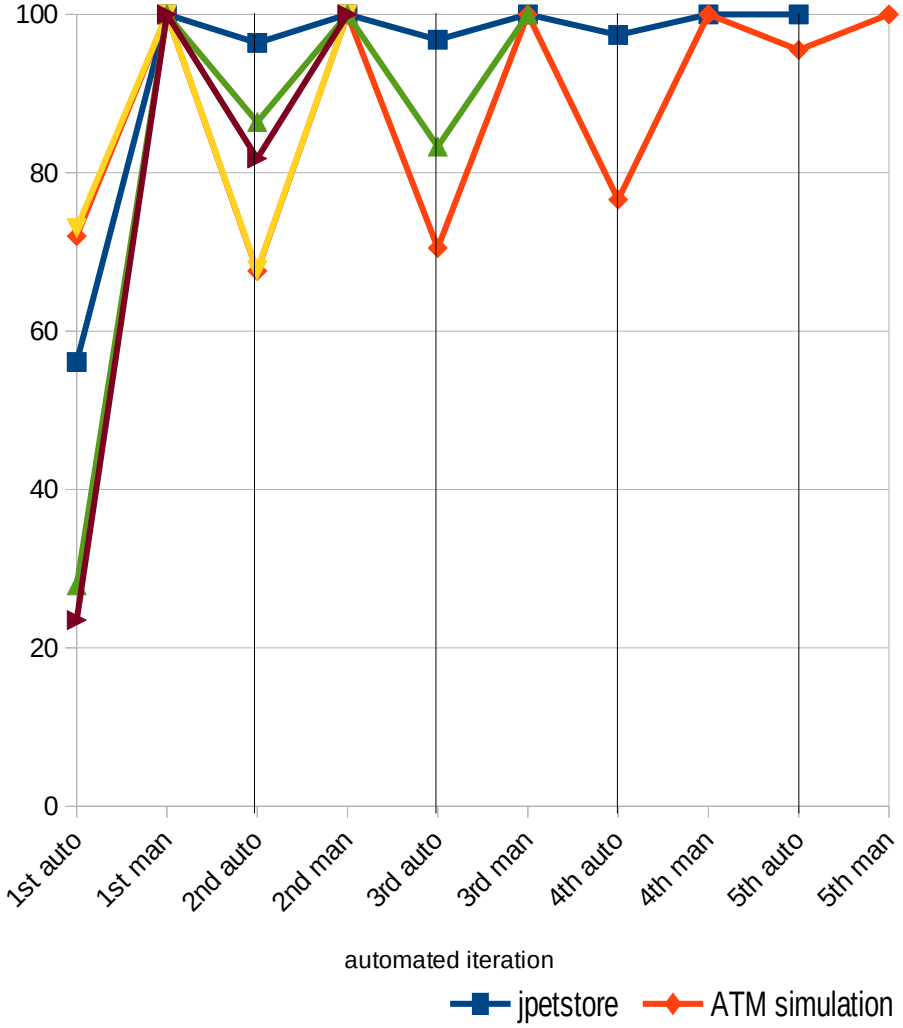
Evaluation – Testdata

- On 5 realistic Java projects
  - Reverse-engineered DFDs
  - Created expected mappings

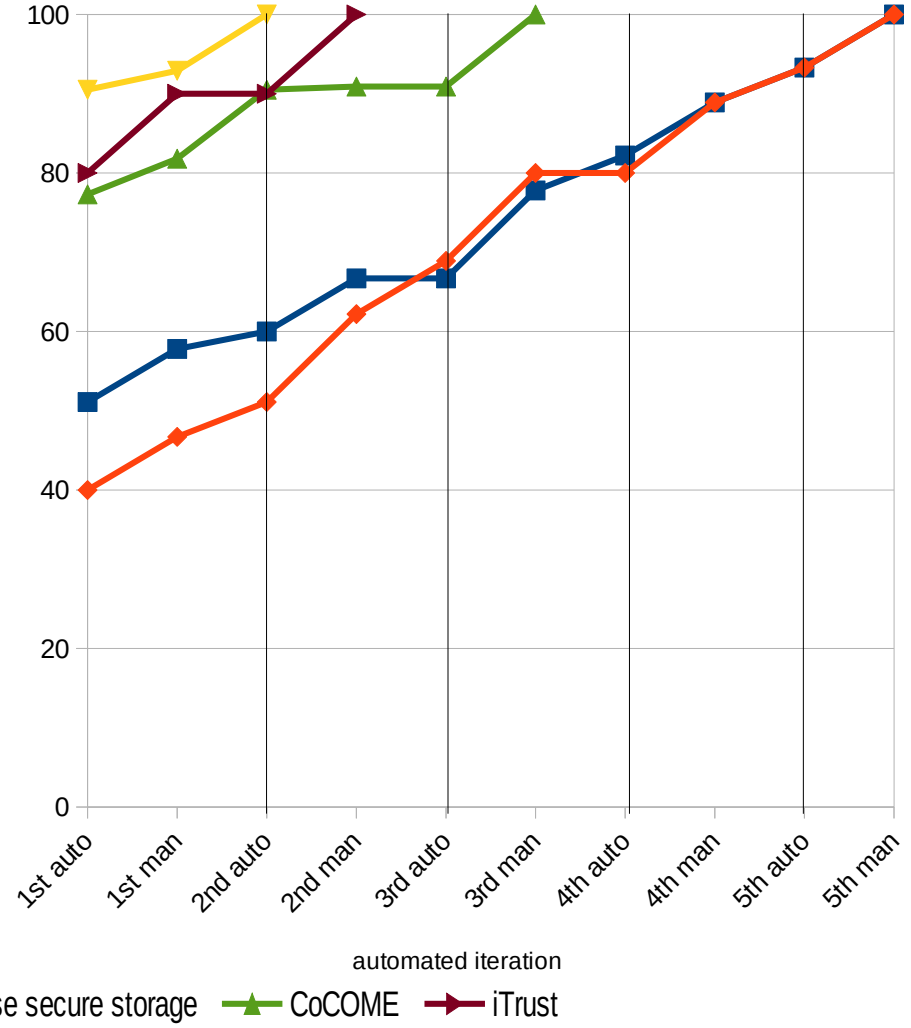
	source code			DFD
project	lloc	classes	methods	elements
jpetstore	1,221	17	277	47
ATM simulation	2,290	57	225	85
Eclipse secure storage	2,900	39	330	41
CoCoME	4,786	120	512	44
iTrust	28,133	423	3,691	31

Evaluation – RQ1

Precision



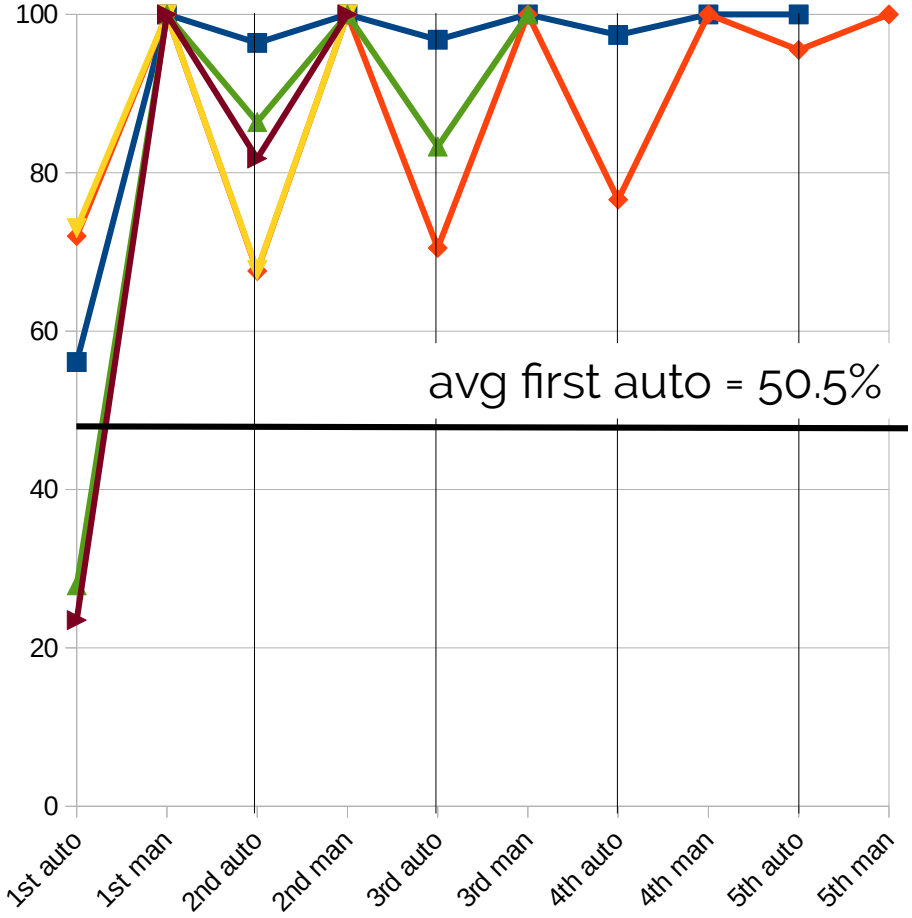
Recall



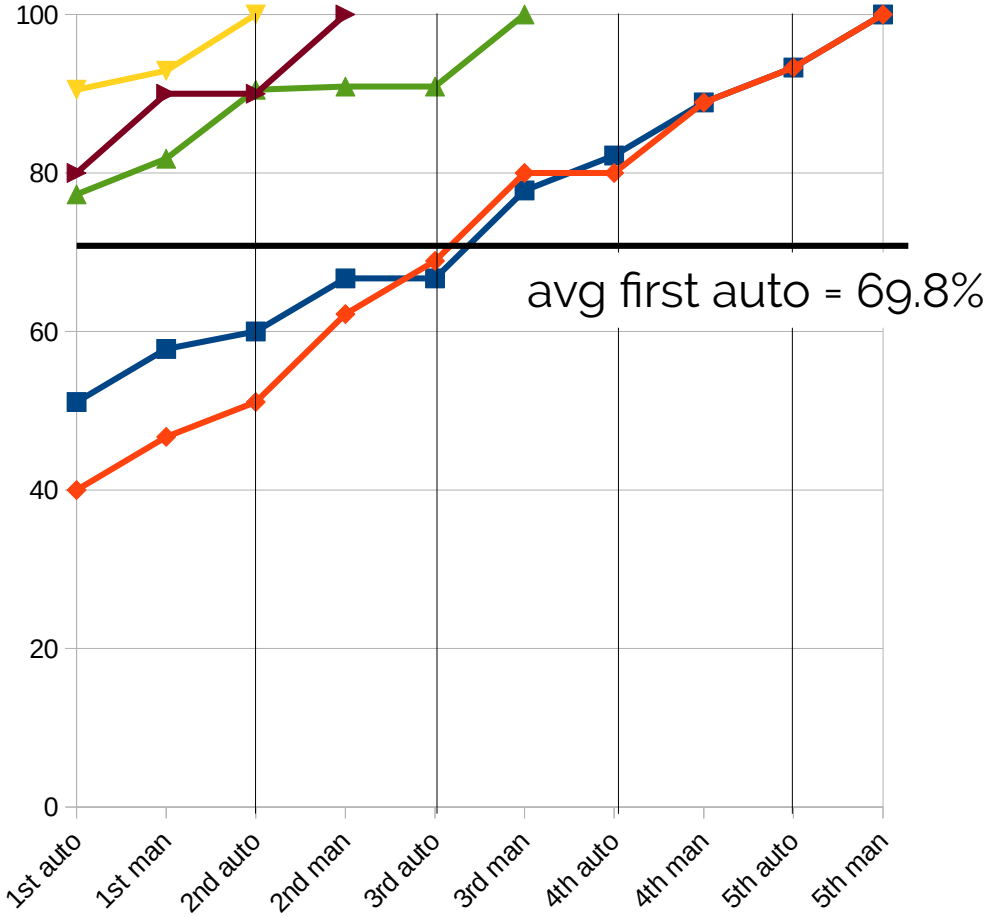
jpetstore | ATM simulation | Eclipse secure storage | CoCOME | iTrust

Evaluation – RQ1

Precision



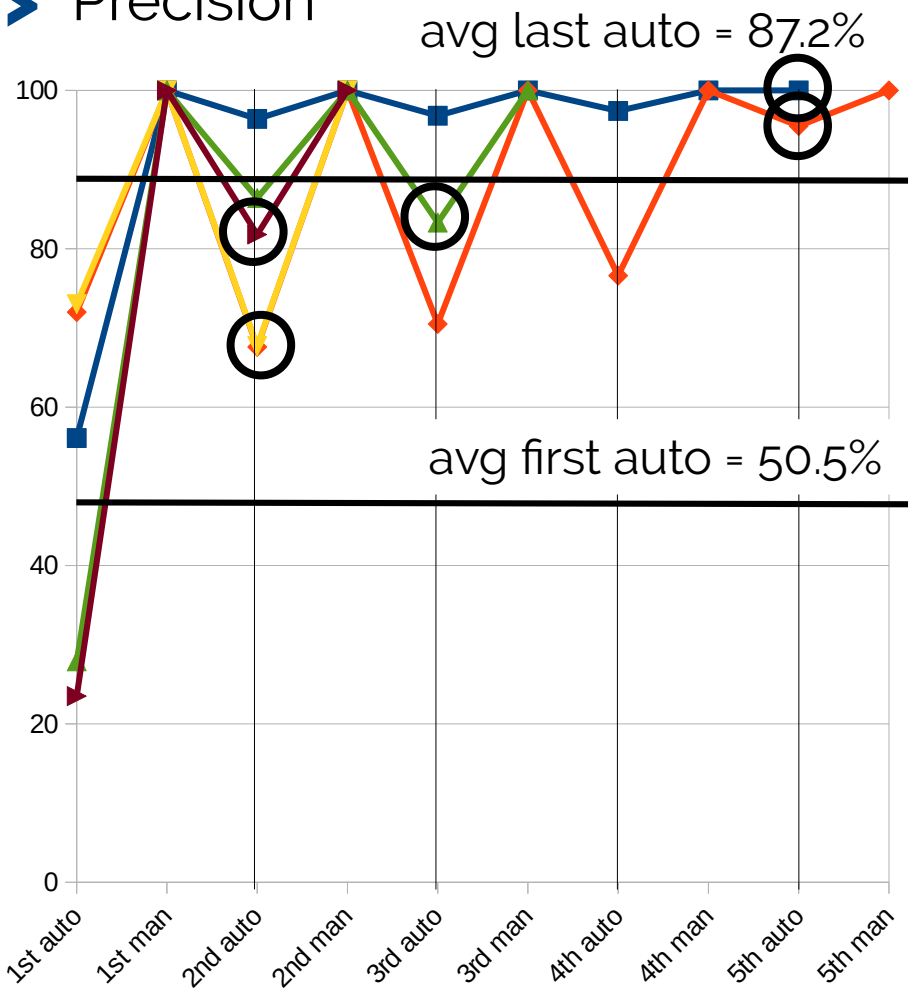
Recall



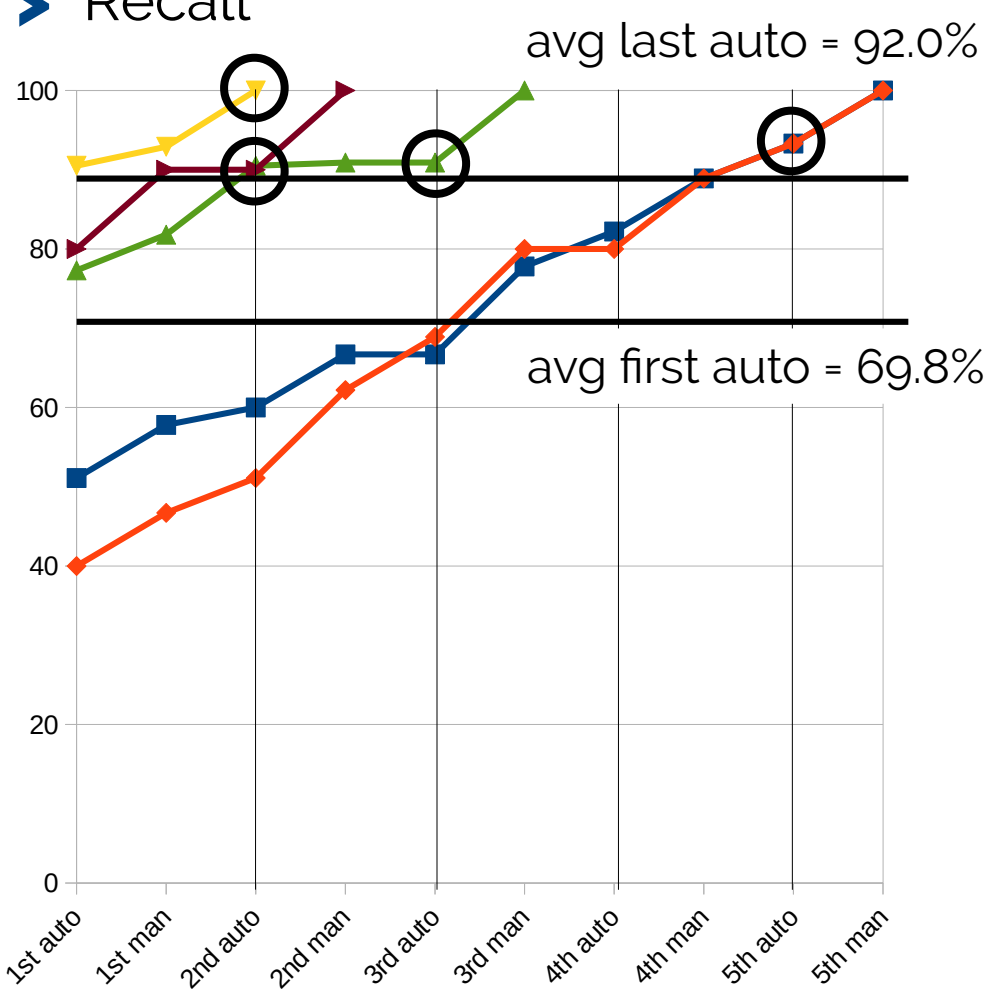
jpetstore | ATM simulation | Eclipse secure storage | CoCOME | iTrust

Evaluation – RQ1

Precision



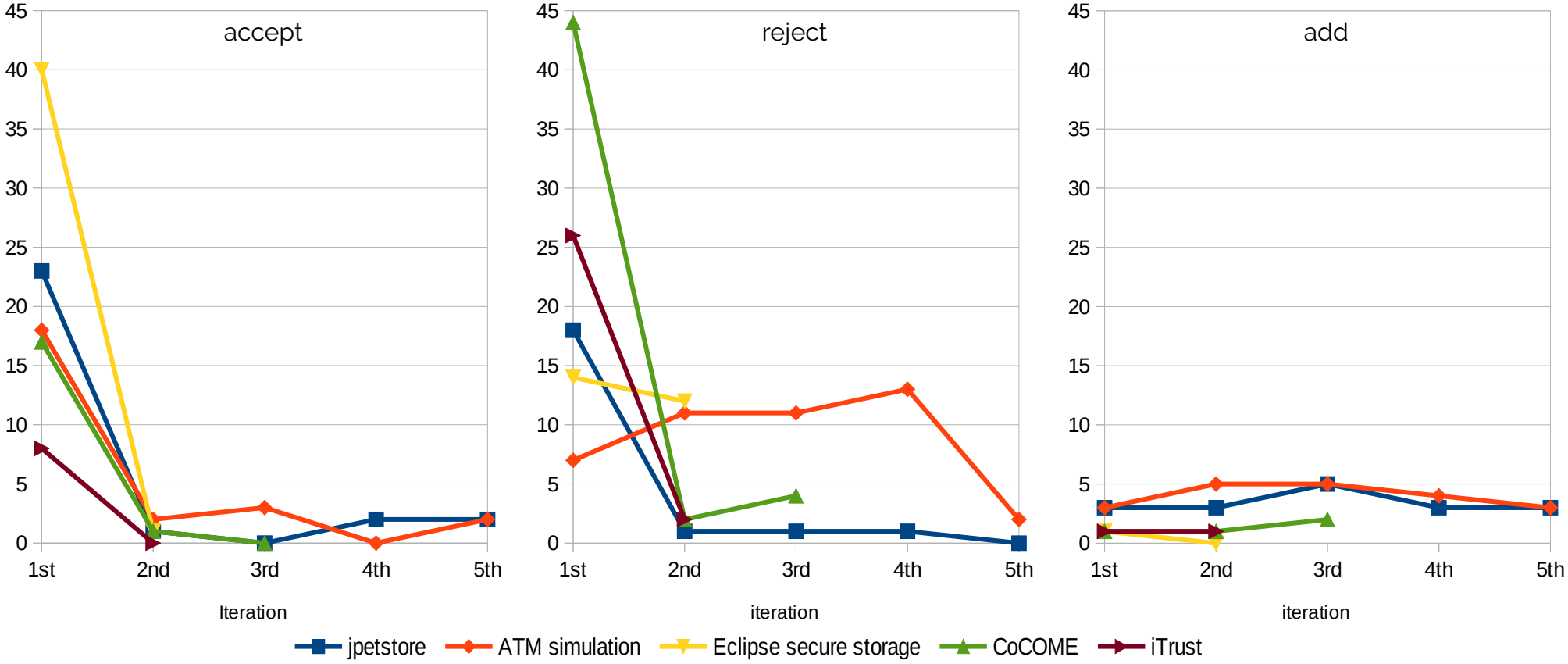
Recall



jpetstore | ATM simulation | Eclipse secure storage | CoCOME | iTrust

Evaluation – RQ2

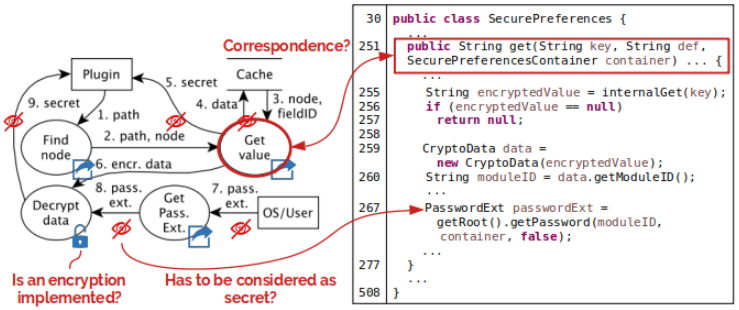
- User actions per iteration:
  - Avg accepts: 7
  - Avg rejects: 9.6
  - Avg adds: 2.6
- Average user impact on recall
  - user: 7.9%
  - automated: 92.1%



Conclusion

Compliance between Models & Code

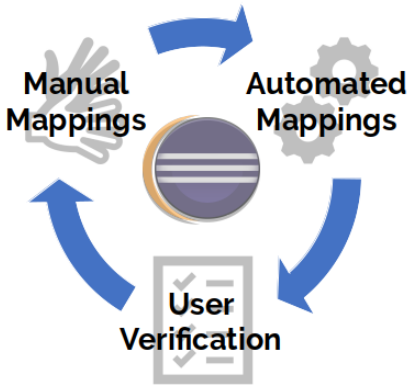
- According to the principle of **security by design**, the system's assets and threats have to be defined in the earliest phases of development.
- Specification of the system (e.g. Eclipse Secure Storage) using a **Security Data Flow Diagram (SecDFD)** [2]




```
30  public class SecurePreferences {
    ...
251   public String get(String key, String def,
      SecurePreferencesContainer container) ... {
      ...
255     String encryptedValue = internalGet(key);
256     if (encryptedValue == null)
257       return null;
258
259     CryptoData data =
          new CryptoData(encryptedValue);
260     String moduleID = data.getModuleID();
        ...
267     PasswordExt passwordExt =
          getRoot().getPassword(moduleID,
          container, false);
        ...
277   }
      ...
508 }
```

Sven Peldszus | Secure Data-Flow Compliance Checks between Models and Code based on Automated Mappings 5

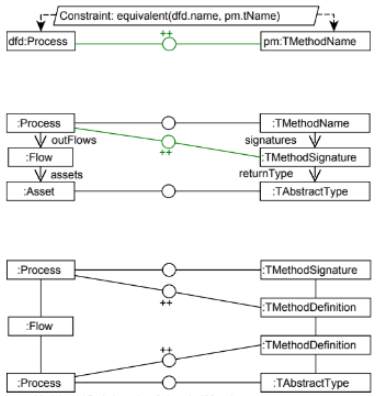
Semi-automated Mappings



Sven Peldszus | Secure Data-Flow Compliance Checks between Models and Code based on Automated Mappings 8

Mapping a DFD to the Program Model

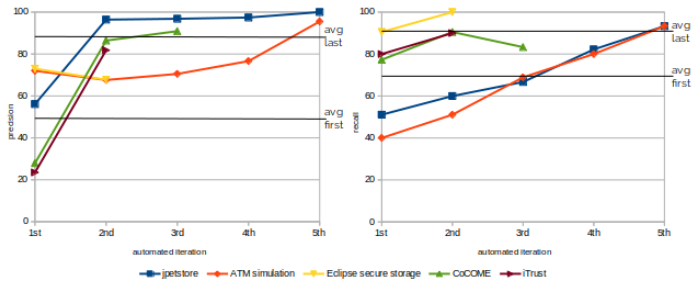
- Iterative mapping
  - 1) Mapping of Names:
    - Using a Heuristic
    - Selecting all with a certainty higher than 70%
  - 2) Mapping of Signatures:
    - Selecting all with at least one parameter or return type mapping
    - Certainty based on number of matches and name matching
  - 3) Mapping of Definitions:
    - Certainty based on signature mappings



Sven Peldszus | Secure Data-Flow Compliance Checks between Models and Code based on Automated Mappings 10

Evaluation – RQ1

- In the first automated iteration
  - Average precision: 50.5%
  - Average recall: 69.8%
- In the final automated iteration
  - Average precision: 87.2%
  - Average recall: 92%



Sven Peldszus | Secure Data-Flow Compliance Checks between Models and Code based on Automated Mappings 17

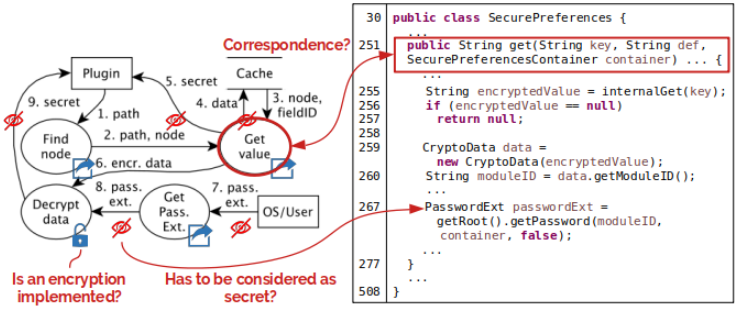
Thank you!

UNIVERSITÄT KOBLENZ · LANDAU

ist Institute for Software Technology

Compliance between Models & Code

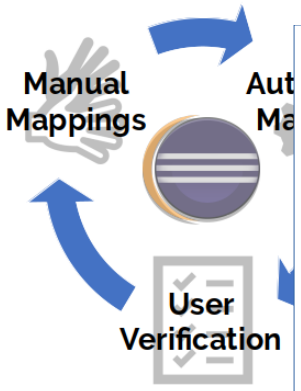
- According to the principle of **security by design**, the system's assets and threats have to be defined in the earliest phases of development.
- Specification of the system (e.g. Eclipse Secure Storage) using a **Security Data Flow Diagram (SecDFD)** [2]




```
30  public class SecurePreferences {
...
251   public String get(String key, String def,
        SecurePreferencesContainer container) ... {
...
255     String encryptedValue = internalGet(key);
256     if (encryptedValue == null)
257       return null;
258
259     CryptoData data =
          new CryptoData(encryptedValue);
260     String moduleID = data.getModuleID();
...
267     PasswordExt passwordExt =
          getRoot().getPassword(moduleID,
          container, false);
...
277   }
...
508 }
```

Sven Peldszus | Secure Data-Flow Compliance Checks between Models and Code based on Automated Mappings 5

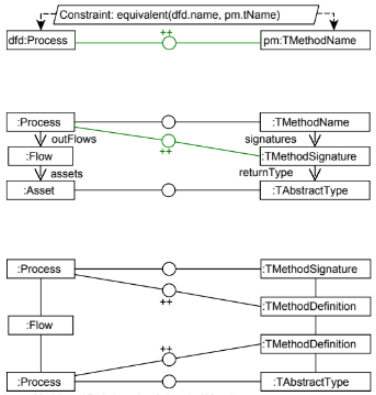
Semi-automated Mappings



Sven Peldszus | Secure Data-Flow Compliance Checks between Models and Code

Mapping a DFD to the Program Model

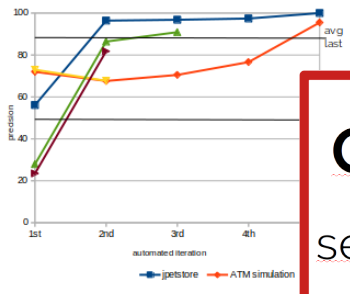
- Iterative mapping
  - 1) Mapping of Names:
    - Using a Heuristic
    - Selecting all with a certainty higher than 70%
  - 2) Mapping of Signatures:
    - Selecting all with at least one parameter or return type mapping
    - Certainty based on number of matches and name matching
  - 3) Mapping of Definitions:
    - Certainty based on signature mappings



Sven Peldszus | Secure Data-Flow Compliance Checks between Models and Code based on Automated Mappings 10

Evaluation – RQ1

- In the first automated iteration
  - Average precision: 50.5%
  - Average recall: 69.8%
- In th
  - A
  - A



Sven Peldszus | Secure Data-Flow Compliance Ch

Visit me at the poster session

Secure Data-Flow Compliance Checks between Models and Code based on Automated Mappings

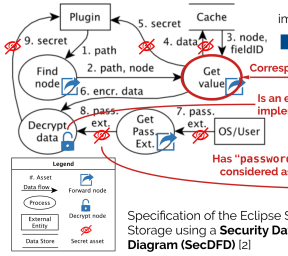
Sven Peldszus[1], Katja Tuma[2], Daniel Strüber[2], Jan Jürjens[1,3], Riccardo Scandariato[2]
1) University of Koblenz-Landau 2) University of Gothenburg and Chalmers University of Technology 3) Fraunhofer ISST

CONTEXT

According to the principle of **security by design**, the system's assets and threats have to be defined in the earliest phases of development.

PROBLEM DESCRIPTION

- Does the implementation comply to the model?
- Do the designed security properties hold?
  - → For checking the compliance we need a mapping between models and code



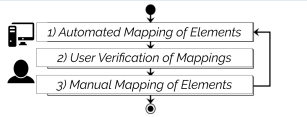
Specification of the Eclipse Secure Storage using a **Security Data Flow Diagram (SecDFD)** [2]

```
30  public class SecurePreferences {
...
251   public String get(String key, String def,
        SecurePreferencesContainer container) ... {
...
255     String encryptedValue = internalGet(key);
256     if (encryptedValue == null)
257       return null;
258
259     CryptoData data = new CryptoData(encryptedValue);
260     String moduleID = data.getModuleID();
...
267     PasswordExt passwordExt = getRoot()
          .getPassword(moduleID, container, false);
...
277   }
...
508 }
```

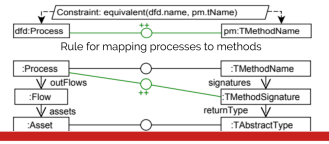
SEMI-AUTOMATED APPROACH

Mappings between SecDFDs and program models for:

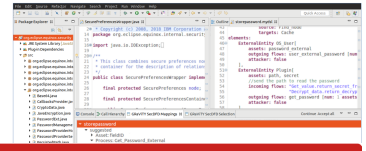
- Compliance checks
  - Convergence
  - Divergence
  - Absence
- Transfer of security labels
  - Security metrics
  - Secure data flow analyses
  - Runtime-monitoring



Automated Mapping of Elements
- Based on **GRaViTY** program model [3]
  - Abstracts details from statement level
  - Detailed type and method information
- Multiple mapping rules and heuristics



User Verification of Mappings and *Manual Mapping of Elements* integrated into the Eclipse IDE
- Accept Mapping → Taken as valid
- Reject Mapping → Never shown again



GET THE TOOL!

secdfd.gravity-tool.org

KOBLENZ · LANDAU GOTHENBURG UNIVERSITY OF TECHNOLOGY Fraunhofer