Requirement As Code: Security Requirements Formalization

Ildar Nigmatullin, University of Toulouse – IRIT – CNRS

Supervisors:

- Sophie Ebersold-Marcaillou, University of Toulouse – IRIT – CNRS, France
- Andrey Sadovykh, Softeam, France
- Nan Messe, University of Toulouse – IRIT – CNRS, France

Agenda

01 Problem statement

02 Idea

03 Why it is helpful

04 Industrial & Practitioners study

05 Limitations and perspectives

Problem Statement

NL ambiguity

Policies standards:

ISO/IEC 27001/27002, NIST, OWASP ASVS change over time

No verification in semi-formal methods

Developer friendlines s

Need of developer-native, executable requirements

Rigorous formal methods

Hard to adopt

What we did?

Analyzed formal, semi-formal and seamless approaches

Came up with seamless methods when Requirements and Code are in the same notation

Research Questions

Can security requirements be formalized using Object-Oriented Programming (OOP) approach combined with seamlesness?

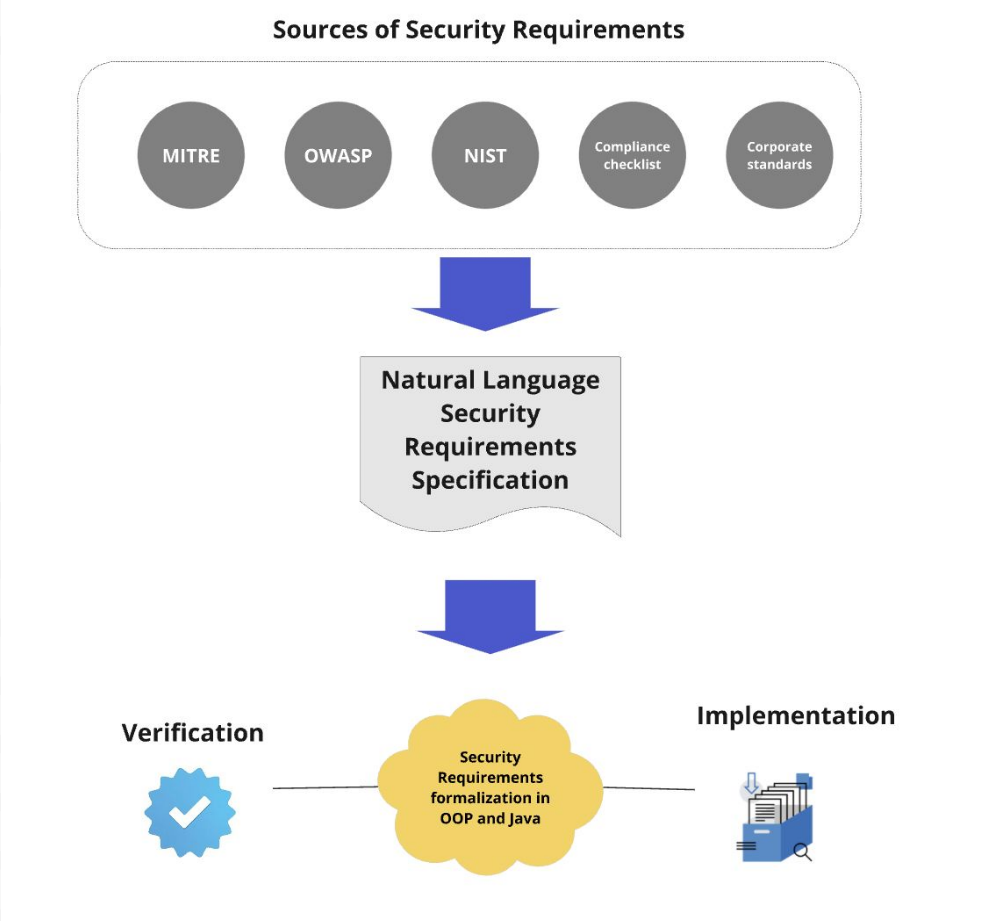
To what extent does OOP and seamlessness help industrial practitioners to formalize security requirements?

Challenges with Security Requirements

They are commonly expressed in **natural language**

Security requirements specifications are often high-level and vague

Difficulty to **interpret**, **analyse**, **verify**, **maintain** and **reuse**



Sources of Security Requirements

MITRE

OWASP

NIST

Compliance checklist

Corporate standards

Natural Language Security Requirements Specification

Verification

Security Requirements formalization in OOP and Java

Implementation

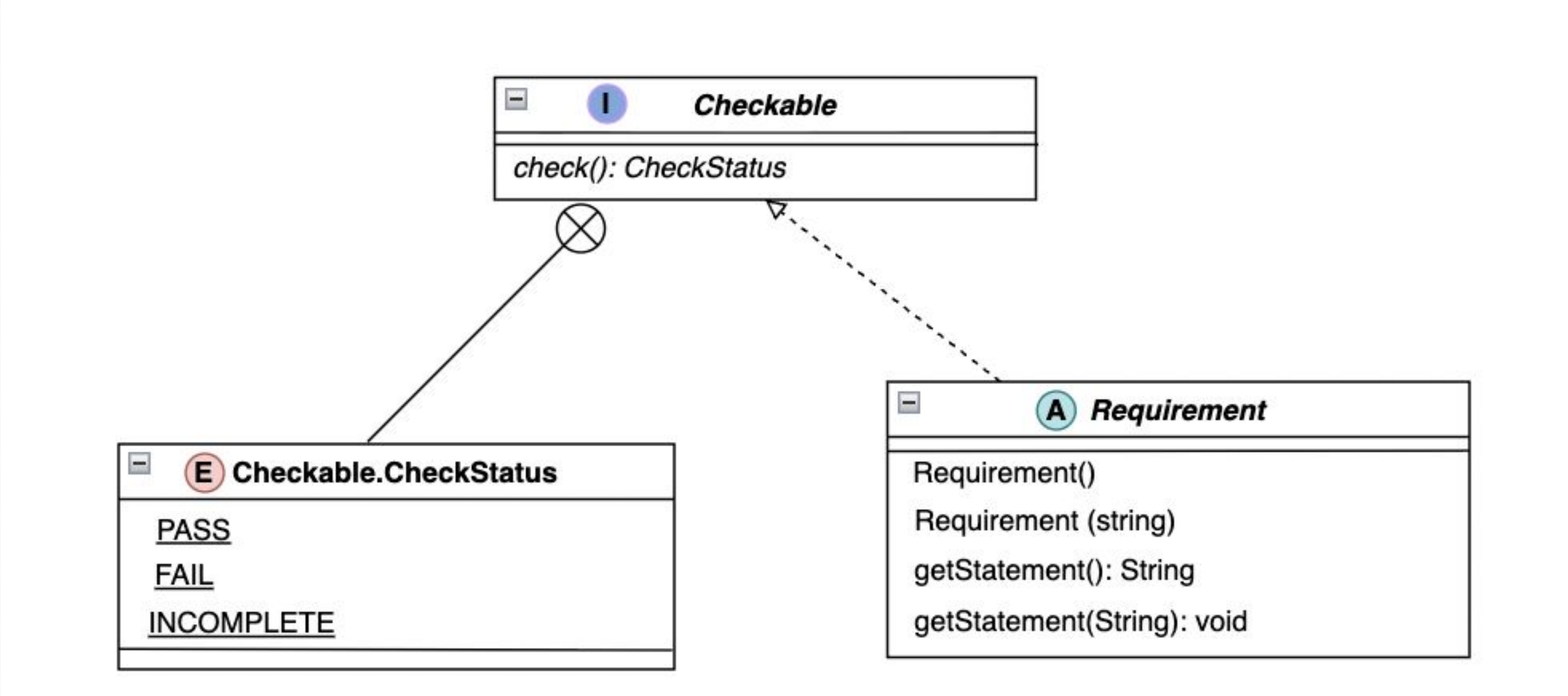
RQCODE – ReQuirements as CODE

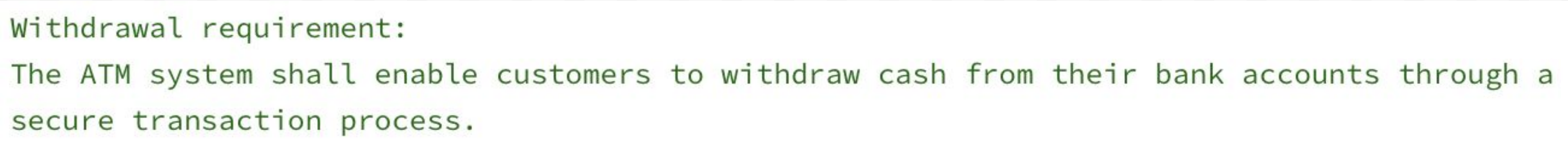
Requirement is OOP class

Java language and environment

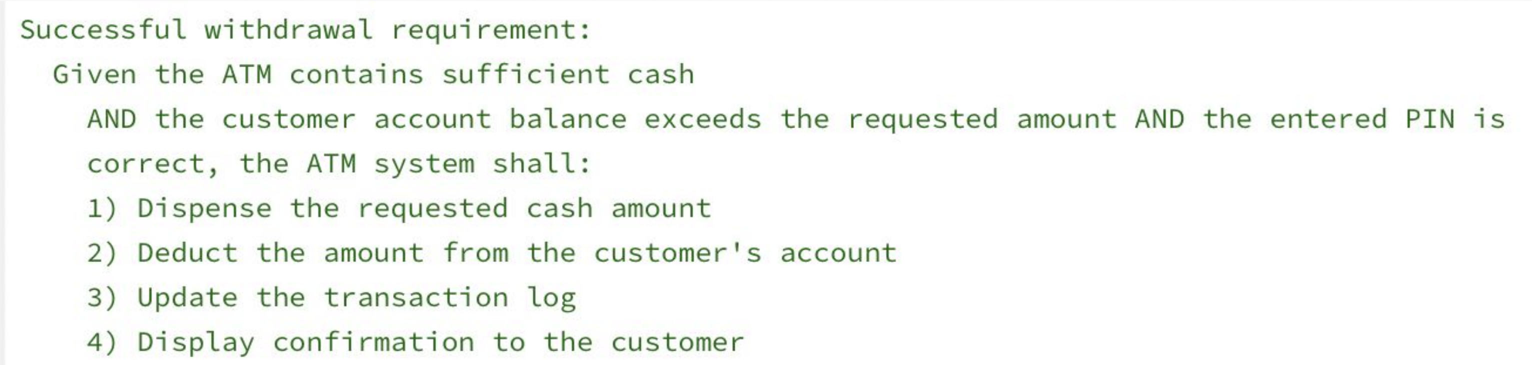
Embedded Verification

Checkable specification

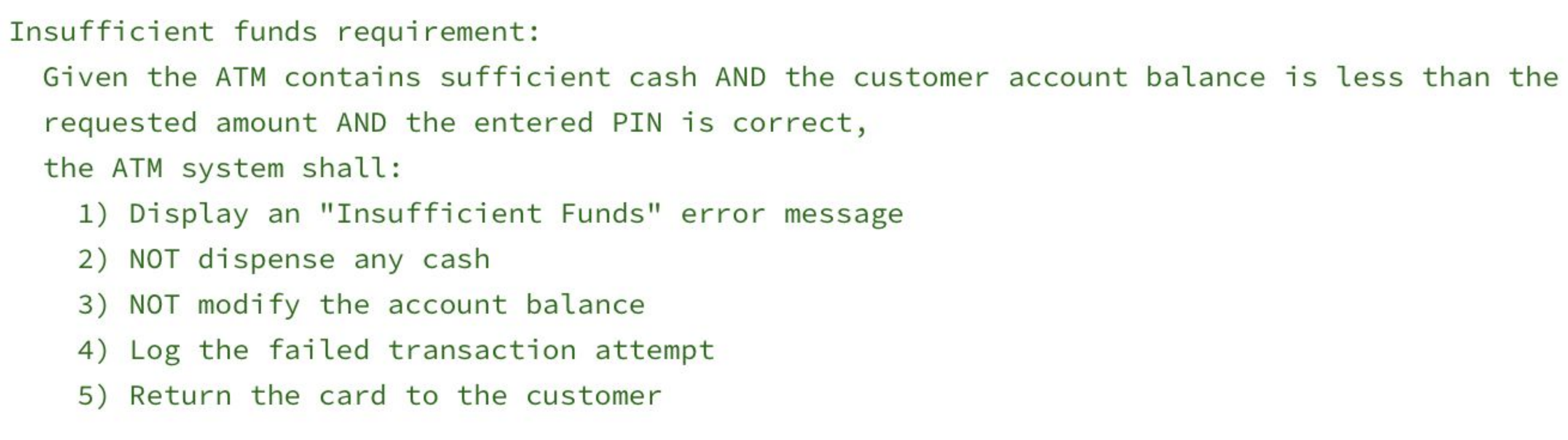




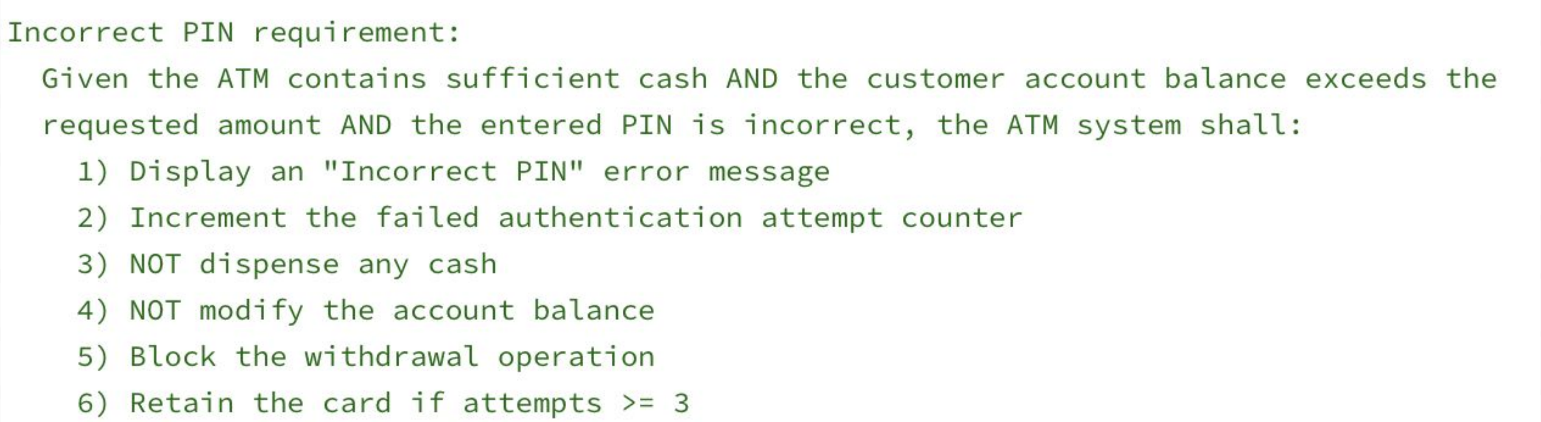
```
Withdrawal requirement:
The ATM system shall enable customers to withdraw cash from their bank accounts through a
secure transaction process.
```



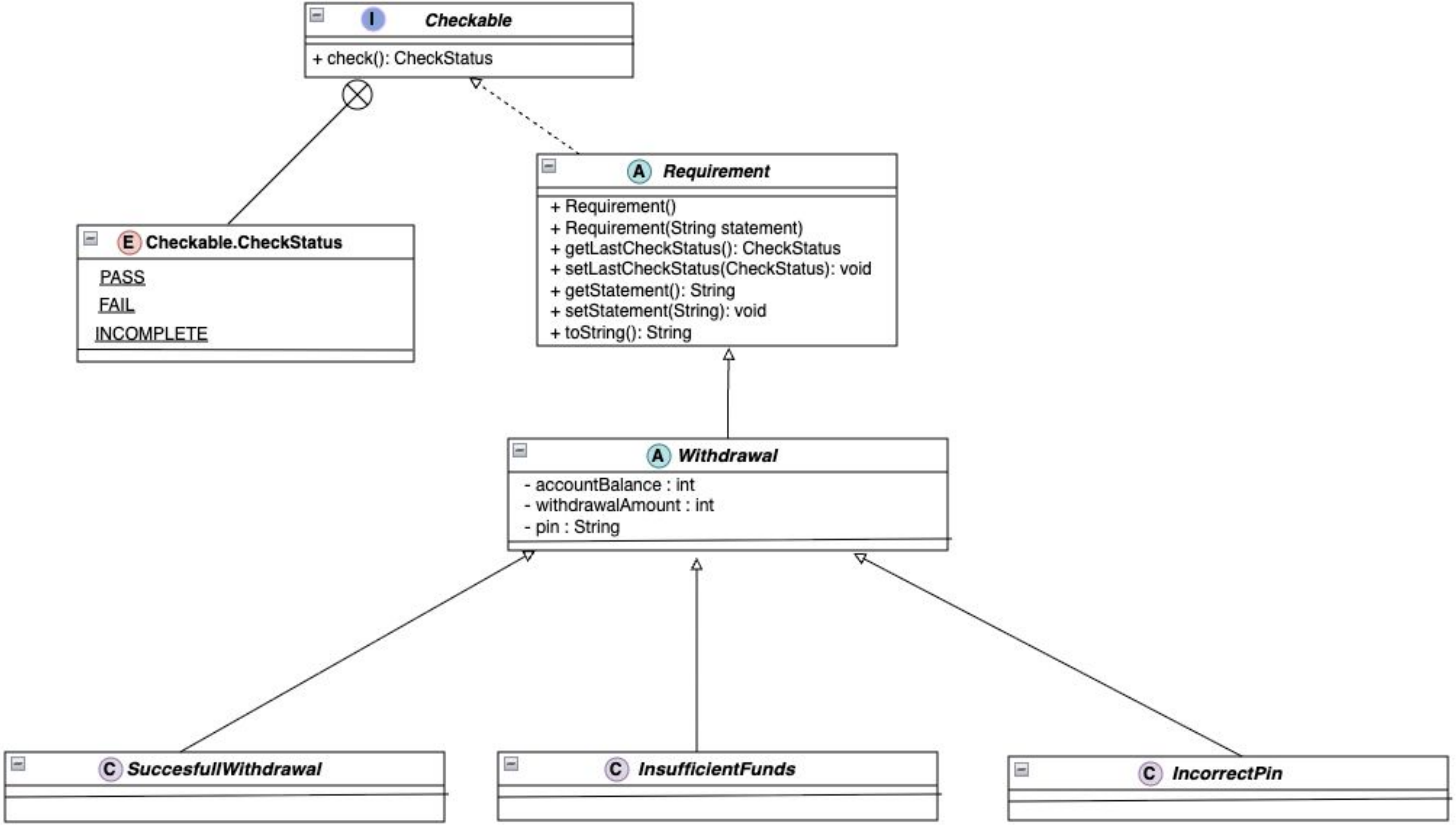
```
Successful withdrawal requirement:
  Given the ATM contains sufficient cash
    AND the customer account balance exceeds the requested amount AND the entered PIN is
    correct, the ATM system shall:
    1) Dispense the requested cash amount
    2) Deduct the amount from the customer's account
    3) Update the transaction log
    4) Display confirmation to the customer
```



```
Insufficient funds requirement:
  Given the ATM contains sufficient cash AND the customer account balance is less than the
  requested amount AND the entered PIN is correct,
  the ATM system shall:
    1) Display an "Insufficient Funds" error message
    2) NOT dispense any cash
    3) NOT modify the account balance
    4) Log the failed transaction attempt
    5) Return the card to the customer
```



```
Incorrect PIN requirement:
  Given the ATM contains sufficient cash AND the customer account balance exceeds the
  requested amount AND the entered PIN is incorrect, the ATM system shall:
    1) Display an "Incorrect PIN" error message
    2) Increment the failed authentication attempt counter
    3) NOT dispense any cash
    4) NOT modify the account balance
    5) Block the withdrawal operation
    6) Retain the card if attempts >= 3
```



I Checkable

+ check(): CheckStatus

E Checkable.CheckStatus

PASS

FAIL

INCOMPLETE

A Requirement

+ Requirement()

+ Requirement(String statement)

+ getLastCheckStatus(): CheckStatus

+ setLastCheckStatus(CheckStatus): void

+ getStatement(): String

+ setStatement(String): void

+ toString(): String

A Withdrawal

- accountBalance : int

- withdrawalAmount : int

- pin : String

C SuccesfullWithdrawal

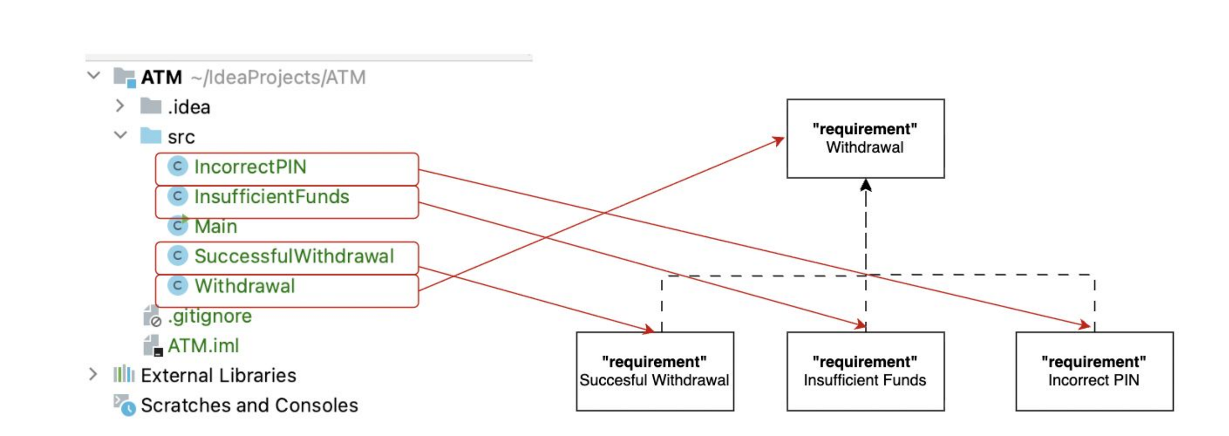
C InsufficientFunds

C IncorrectPin

Why is it helpful?

Requirements Reusability

Requirement s traceability



Industrial study

- **Context:** industrial partner managing Security Technical Implementation Guides (STIGs) for Windows 10.
- **Problem**: large set of NL security checks and PowerShell scripts, difficult to maintain and reuse.
- **Approach:** encode selected STIG findings as RQCODE classes,
- **Outcome:** demonstrates feasibility of applying RQCODE at scale and better control over evolution of hardening rules.

Industrial study

- **Context:** industrial partner managing Security Technical Implementation Guides (STIGs) for Windows 10.
- **Problem**: large set of NL security checks and PowerShell scripts, difficult to maintain and reuse.
- **Approach:** encode selected STIG findings as RQCODE classes,
- **Outcome:** demonstrates feasibility of applying RQCODE at scale and better control over evolution of hardening rules.

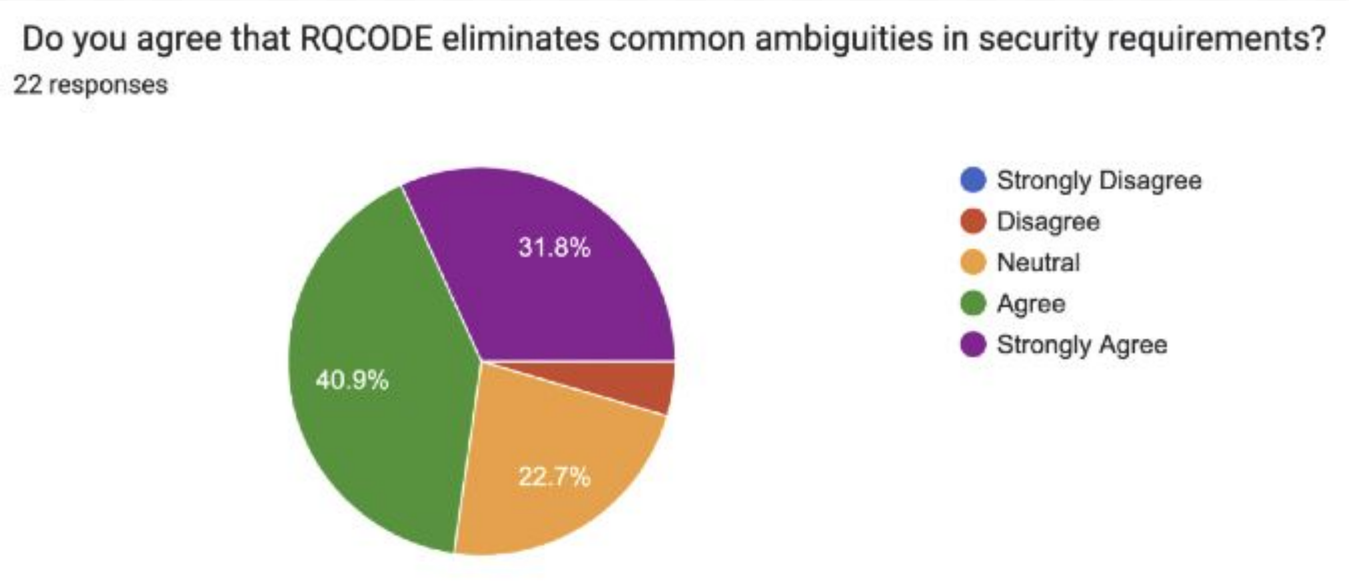
Software practitioners study

- Participants: software engineering students and practitioners (n ≈ 22 for survey phase).
  - short tutorial on RQCODE and password policy case,
  - hands-on assignment implementing security requirements in RQCODE,
  - post-workshop survey on perceived benefits and effort.
- Measures: perceived ambiguity reduction, understandability, verification and testing support.
- Analysis: Likert-scale responses and descriptive statistics.

Software practitioners study

- Majority of respondents Agree or Strongly Agree that RQCODE:
  - helps eliminate common ambiguities in security requirements,
  - makes Java-based security requirements easier to understand,
  - makes it easier to verify whether security requirements are satisfied,
  - simplifies the process of testing security requirements.
- No respondents selected Strongly Disagree; only a small minority selected Disagree.

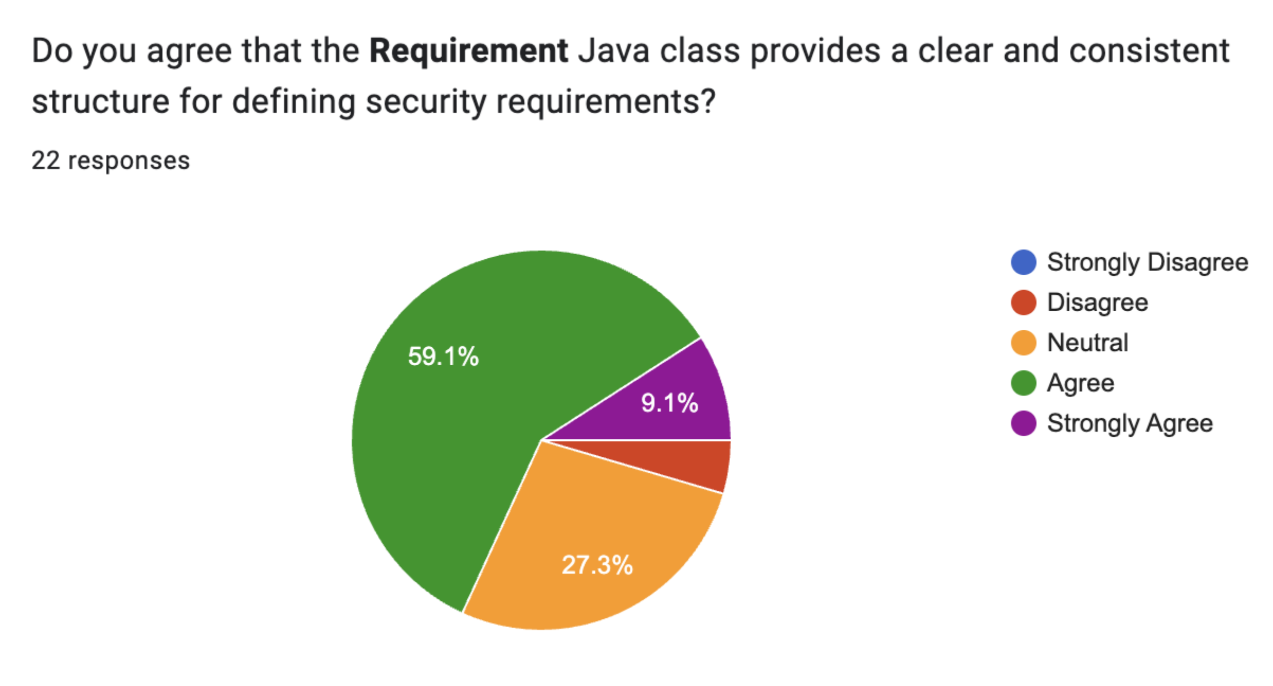
Software practitioners study (1)



Do you agree that RQCODE eliminates common ambiguities in security requirements?

22 responses

Software practitioners study (2)



Do you agree that the **Requirement** Java class provides a clear and consistent structure for defining security requirements?

22 responses

Limitations

- learning curve for teams unfamiliar with Java or requirements-as-code concepts,
- current focus on a subset of security requirements (coverage still limited),
- dependency on the Java ecosystem and tooling.

Perspectives

- **Short-term perspectives:**
  - extend requirement libraries (access control, logging, audit, traceability),
  - improve IDE support and visualisation of requirement relationships,
  - refine evaluation with additional industrial settings.
- **Long-term perspectives:**
  - explore multi-language implementations (e.g., Python, TypeScript),
  - investigate user interface development for RQCODE.
  - Investigate API integration

Thank you!