Is Modeling Access Control Worth It?

David Basin[1] Juan Guarnizo[2] Srđan Krstić[1] Hoang Nguyen[1] Martín Ochoa[2]

1

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

2

Zürcher Hochschule
für Angewandte Wissenschaften

zh aw School of Engineering

CCS 2023
Copenhagen, Denmark

Motivation

Forbes

TECH

4.6 Million Snapchat Usernames And Phone Numbers Captured By API Exploit

Anthony Wing Kosner Former Contributor

Quantum of Content and innovations in user experience

Jan 1, 2014, 01:51am EST

WIRED BACKCHANNEL BUSINESS CULTURE GEAR IDEAS SCIENCE SECURITY MERCH BLACK FRIDAY

LILY HAY NEWMAN SECURITY MAY 24, 2019 6:49 PM

Hack Brief: 885 Million Sensitive Financial Records Exposed Online

Real estate giant First American left Social Security numbers, tax documents, and more publicly available.

WIRED BACKCHANNEL BUSINESS CULTURE GEAR IDEAS SCIENCE SECURITY MORE

ANDY GREENBERG SECURITY JAN 12, 2021 11:43 AM

An Absurdly Basic Bug Let Anyone Grab All of Parler's Data

The "free speech" social network also allowed unlimited access to every public post, image, and video.

THE ZERO HACK

HOME HACKS / SECURITY MAKING MONEY ONLINE TECH TRICKS DIGITAL MARKETING GET IN TOUCH

Hacking Facebook Pages

BY LAXMAN MUTHIYAH / AUG 26, 2015 / HACKS / SECURITY

APIsecurity.io By 42Crunch

Issue 116: Facebook and Parler API vulnerabilities, clairvoyance

January 14, 2021

Vulnerability: Facebook

Pouya Darabi found an API vulnerability in Facebook that allowed him to create posts on other users' pages. The posts were not popping up in the newsfeed, but they were visible and looked legitimate to anyone who would have accessed them through a direct link.

OWASP’s Top Vulnerability

A01:2021 – Broken Access Control

Factors

CWEs Mapped	Max Incidence Rate	Avg Incidence Rate	Avg Weighted Exploit	Avg Weighted Impact	Max Coverage	Avg Coverage	Total Occurrences	Total CVEs
34	55.97%	3.81%	6.92	5.93	94.55%	47.72%	318,487	19,013

Typical causes: misconfiguration of mechanisms or their incorrect implementation

Access Control (AC)

Determines whether actions are authorized and **prevents** unauthorized actions.

Access Control (AC)

Determines whether actions are authorized and **prevents** unauthorized actions.

Authorization policy: Free users can join an event, if the event is public.

Access Control (AC)

Determines whether actions are authorized and **prevents** unauthorized actions.

Authorization policy: Free users can join an event, if the event is public.

Enforcement mechanism: Suppressing the action, rolling back the state, and throwing a (security) exception.

Access Control (AC)

Determines whether actions are authorized and **prevents** unauthorized actions.

Authorization policy: Free users can join an event, if the event is public.

Least privilege: each user can only perform actions that are necessary to carry out their business role

Enforcement mechanism: Suppressing the action, rolling back the state, and throwing a (security) exception.

Complete mediation: every action is checked for authority

Implementing AC: Code-centric approach

Security Requirement Free users can join an event, if the event is public.

Design (informal)

Implementation

```
def join_event(id):
  event = Event.query.get(id)

  event.requesters.append(current_user)
  db.session.commit()
```

Implementing AC: Code-centric approach

Security Requirement

Free users can join an event, if the event is public.

Design (informal)

	VISITOR	...	FREEUSER
...	...	...	...
join_event	✘	...	✓ (public)
...	...	...	...

Implementation

```
def join_event(id):
  event = Event.query.get(id)

  event.requesters.append(current_user)
  db.session.commit()
```

Implementing AC: Code-centric approach

Security Requirement

Free users can join an event, if the event is public.

Design (informal)

	VISITOR	...	FREEUSER
...	...	...	...
join_event	✘	...	✓ (public)
...	...	...	...

Implementation

```
@login_required
@roles_accepted(FREEUSER)
def join_event(id):
  event = Event.query.get(id)
  if not event.private:
    event.requesters.append(current_user)
    db.session.commit()
  else
    raise SecurityException(“Not allowed")
```

Implementing AC: Code-centric approach

Security

Design (i

Impleme

public.

Problems:

- Implementation process is error-prone
- Does not scale with the application size
- Difficult to maintain and evolve

```
else
   raise SecurityException(“Not allowed")
```

Implementing AC: Model-driven approach

Security

Design (f

public.

Model-driven development:

- Proposed more than two decades ago
- Design cross-cutting system aspects
- … using formal system models, and then
- generate correct-by-design code

Implementation

Implementing AC: Model-driven approach

Security Requirement Free users can join an event, if the event is public.

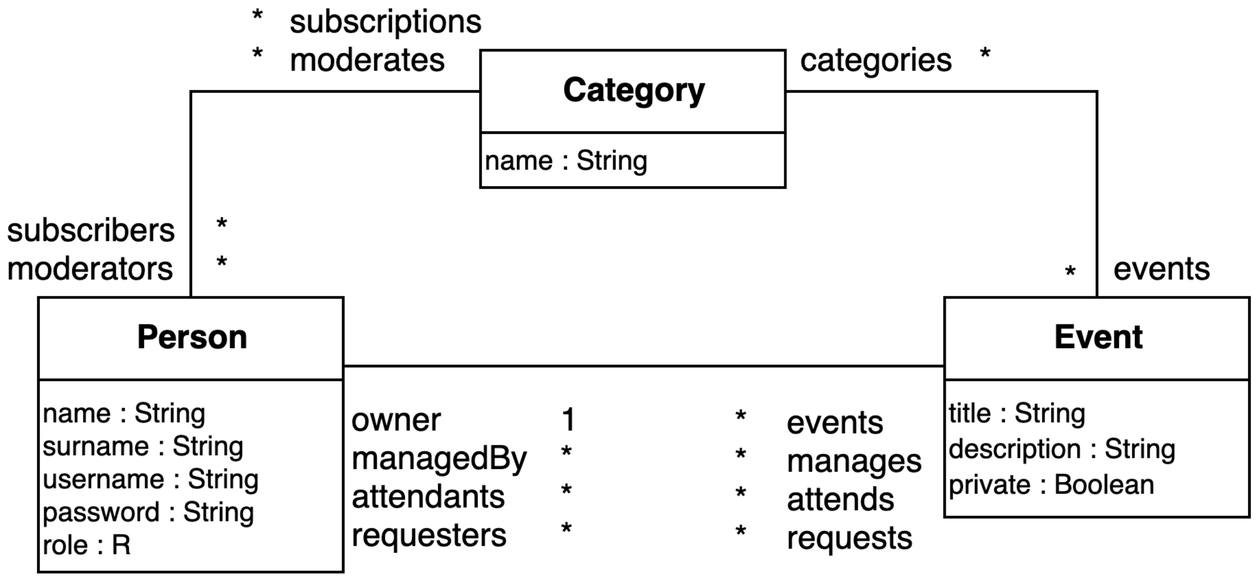
Design (formal)

Implementation

Implementing AC: Model-driven approach

Security Requirement Free users can join an event, if the event is public.

Design (formal) Data model

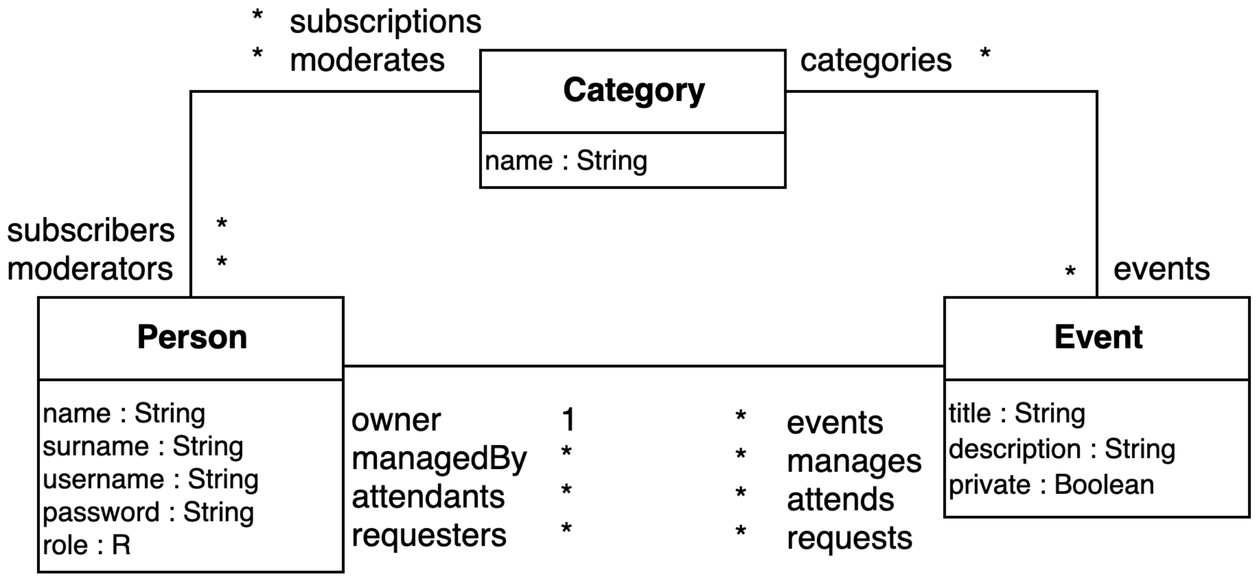


Implementation

Implementing AC: Model-driven approach

Security Requirement Free users can join an event, if the event is public.

Design (formal) Data model



Security model

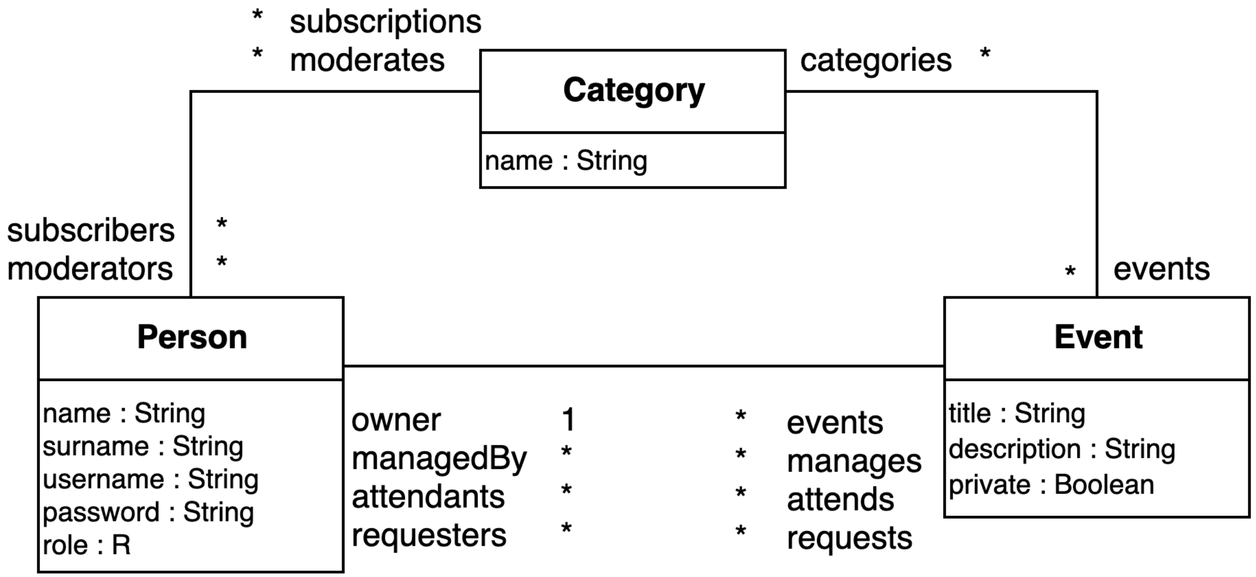
{(FREEUSER,
add(Event, requesters),
not self.private), … }

Implementation

Implementing AC: Model-driven approach

Security Requirement Free users can join an event, if the event is public.

Design (formal) Data model



Security model

{(FREEUSER,
add(Event, requesters),
not self.private), … }

Implementation Automatically generated from the models.

Is modeling AC worth it?

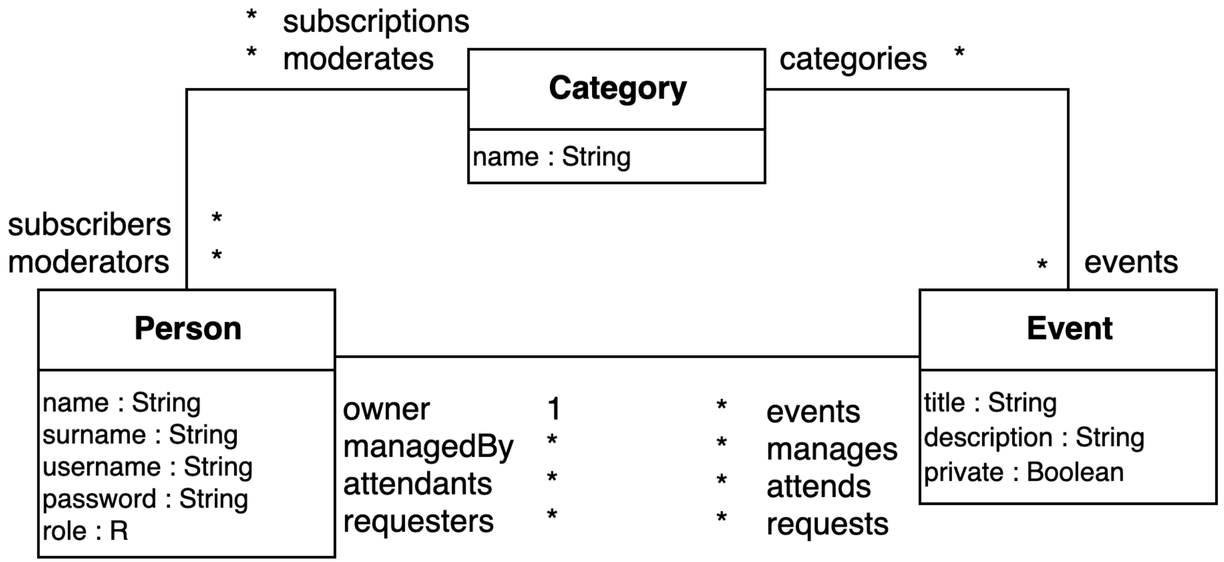
(No empirical study to date)

Let’s do a study!

- 95 developers
  - Students from a security engineering course at ETH Zürich
- Implement AC twice:
  - Code-centric approach with Flask/Python (and other libraries)
  - Model-driven approach with ActionGUI (AG) tool
- Simple web application with complex, realistic AC requirements
  - Focus on implementing AC, other aspects provided

Study Design: Project requirements

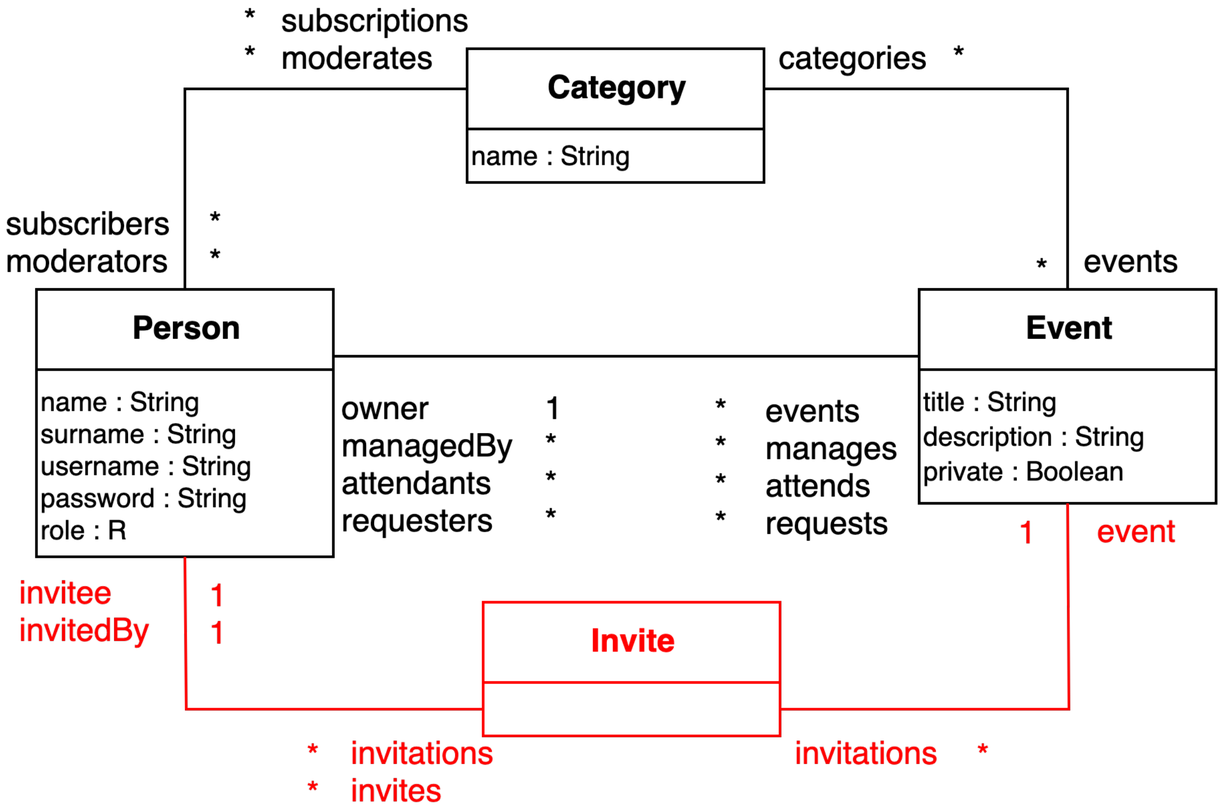
Event management web application



- A simplified version of the typical event management platforms (e.g., Meetup or Facebook events)
- Roles: FREEUSER, PREMIUMUSER, MODERATOR, ADMIN
- Non-trivial security requirements
  - Example: Only an attendee can read a private event's attendees and managers

Study Design: Evolved project requirements

Event management web application



- Extended to support explicit invitations
- Roles remain the same, but many permissions need to be adjusted
  - Example: User invited to a private event can read its attendees and managers

Research Questions

[**LP**] Can modeling AC help achieve least privilege?

[**CM**] Can modeling AC help achieve complete mediation?

[**T** & **LoC**] What effort (time and LoC) is needed to implement, repair, and evolve security requirements?

[**PK**] Is implementing (resp., designing) AC more effective given prior knowledge: design (resp., reference implementation)?

[**MD**] What are the most difficult aspects of AC to implement (resp., specify)?

Research Questions

[**LP**] Can modeling AC help achieve least privilege?

[**CM**] Can modeling AC help achieve complete mediation?

[**T** & **LoC**] What effort (time and LoC) is needed to implement, repair, and evolve security requirements?

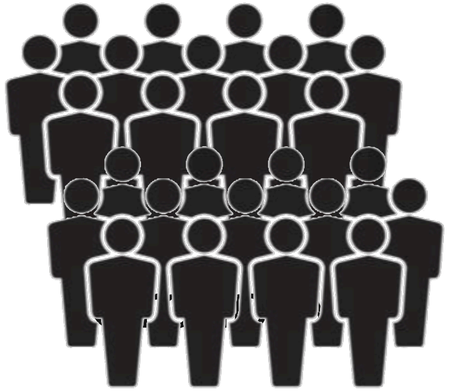
[**PK**] Is implementing (resp., designing) AC more effective given prior knowledge: design (resp., reference implementation)?

[**MD**] What are the most difficult aspects of AC to implement (resp., specify)?

Study Design: Organization

Phase 0 - “Prepare“

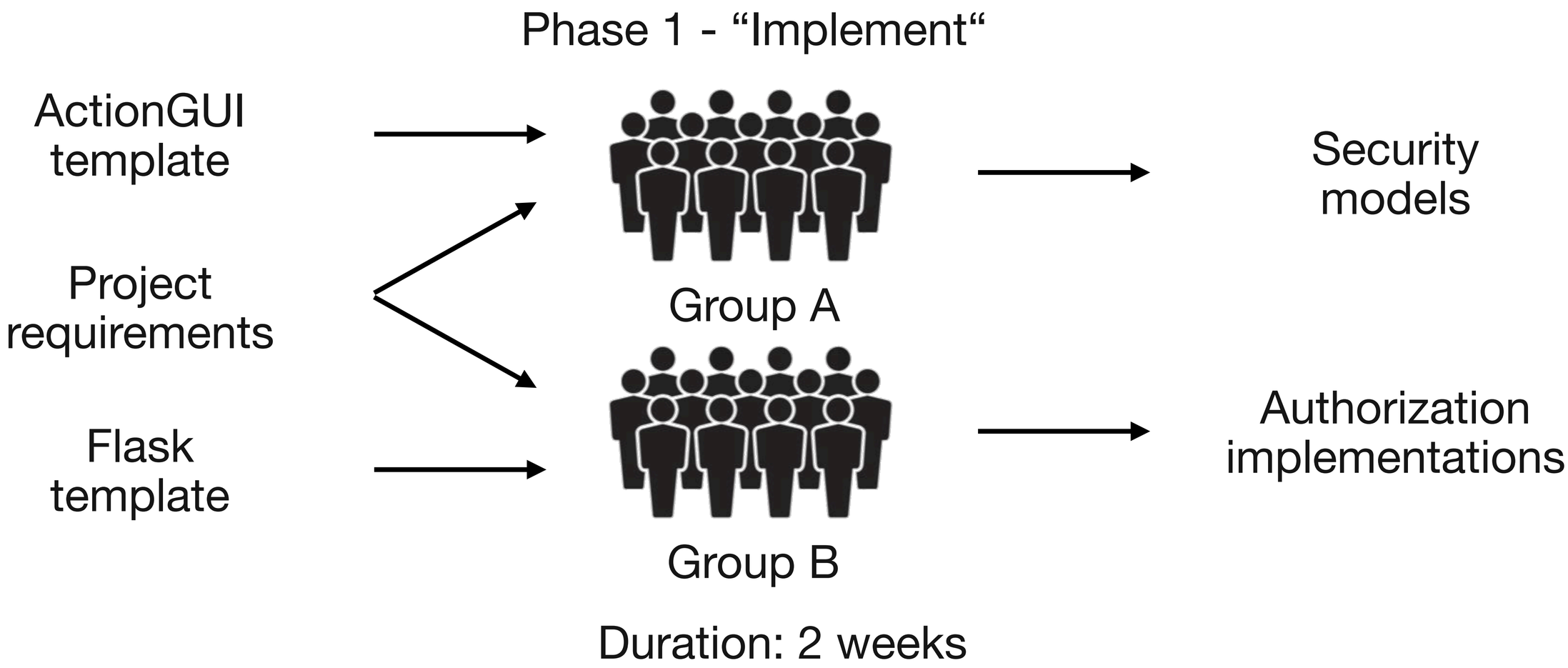
Learning materials
(ActionGUI & Flask)



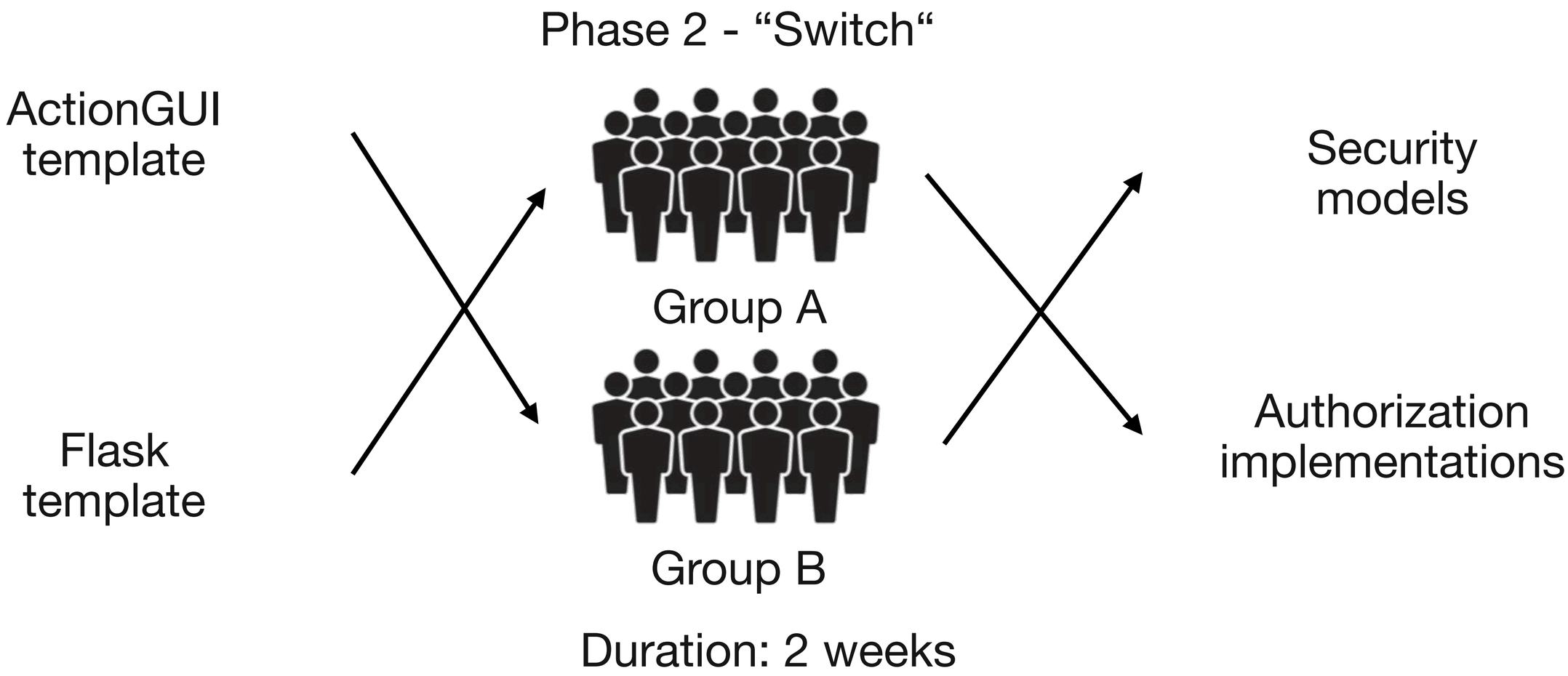
Students

Duration: 6 weeks

Study Design: Organization



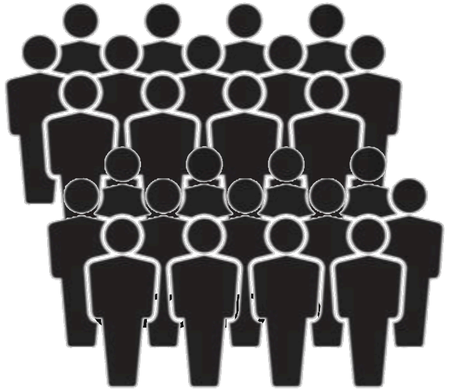
Study Design: Organization



Study Design: Organization

Phase 3 - “Repair“

ActionGUI and Flask security tests



Students

Repaired models & implementations

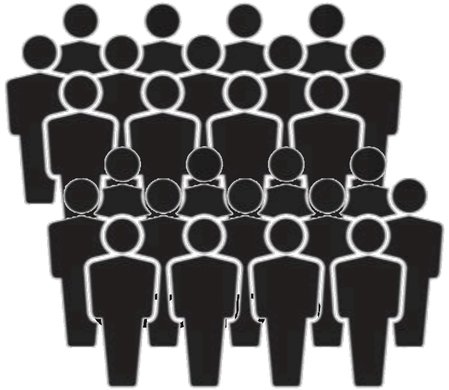
Duration: 1 week

Study Design: Organization

Phase 3 - “Repair“

ActionGUI and Flask security tests

> 1k tests systematically check both authorized and unauthorized actions



Students

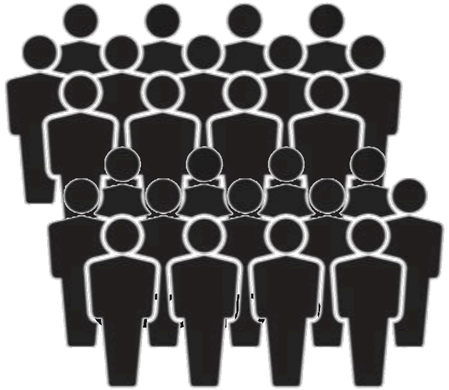
Repaired models & implementations

Duration: 1 week

Study Design: Organization

Phase 4 - “Evolve“

Evolved project requirements



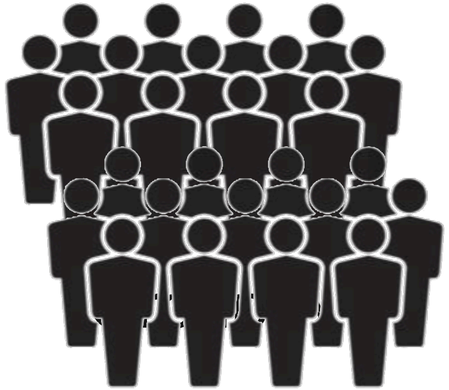
Students

Evolved models & implementations

Duration: 1 week

Study Design: Organization

Phase 5 - “Survey“

Questionnaire →  → Anonymous answers

Students

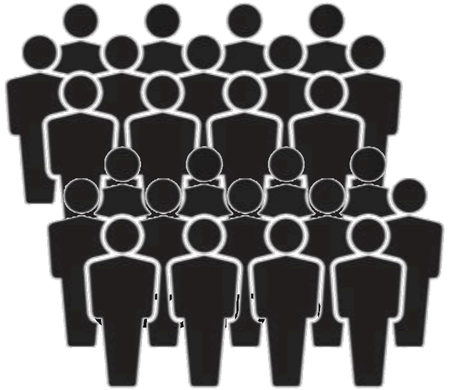
Duration: 1 week

Study Design: Organization

Phase 5 - “Survey“

Questionnaire

20 questions on experiences and opinions of the participants, as well as the time needed for each phase



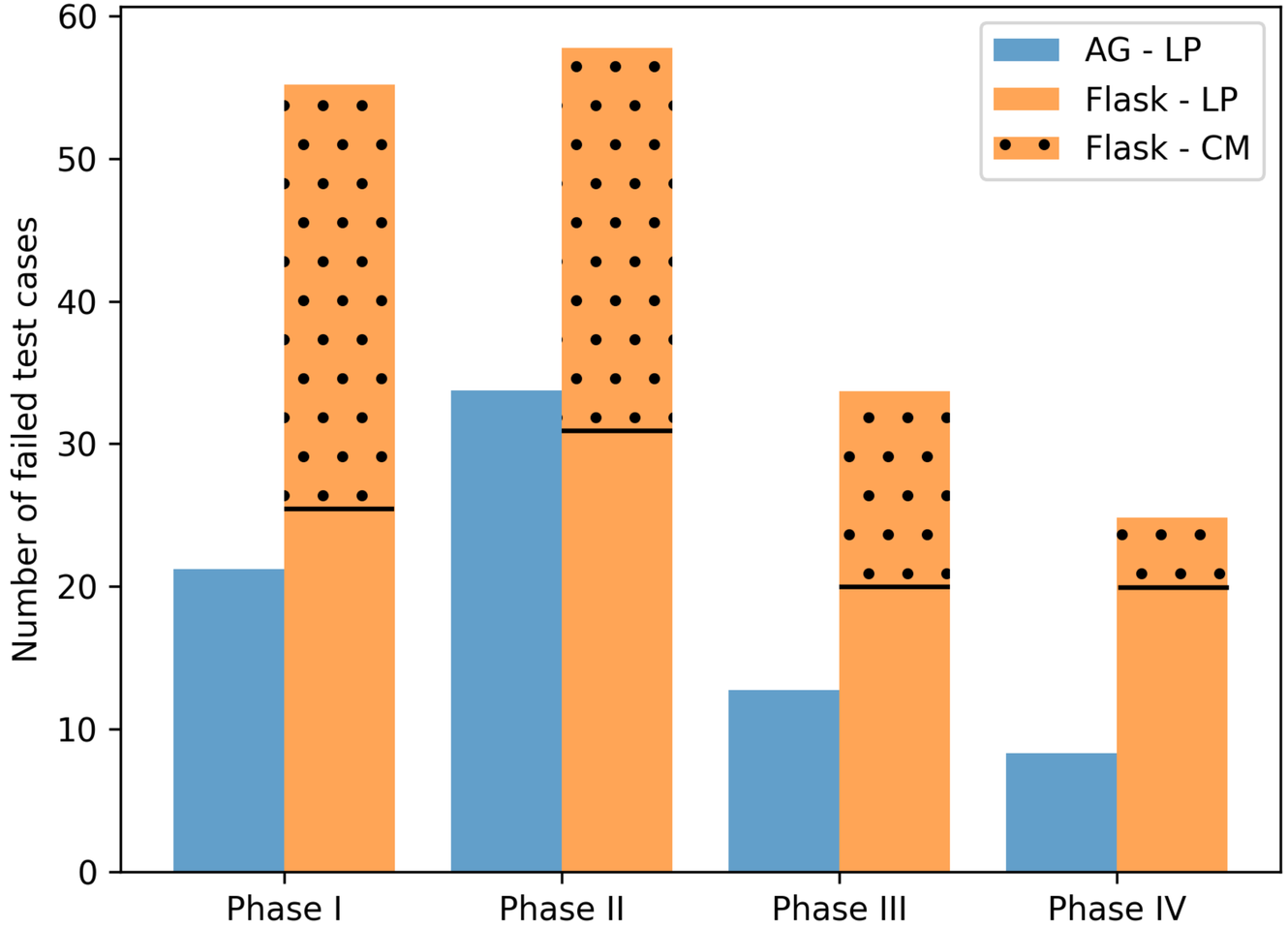
Students

Anonymous answers

Duration: 1 week

Results - Principles [LP & CM]

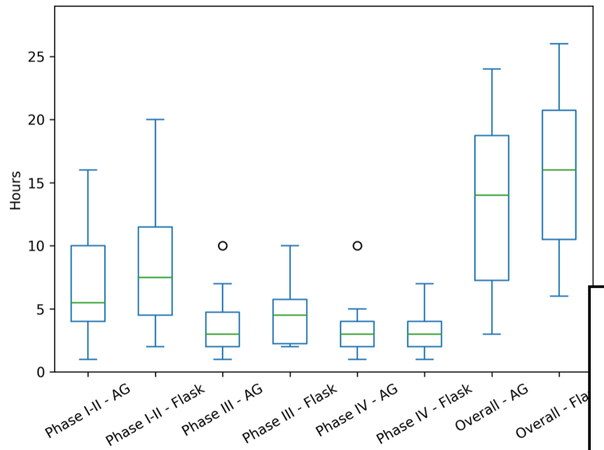
Number of failed cases by phase and violated principle (LP vs CM)

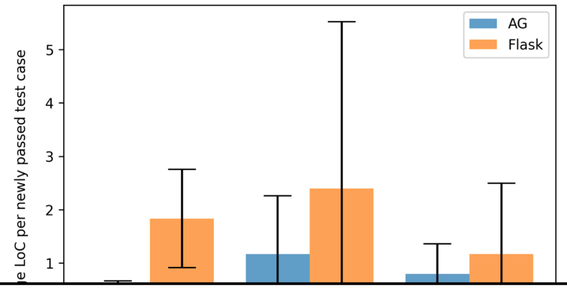


Other Results

Results - Efficiency [T & LoC]

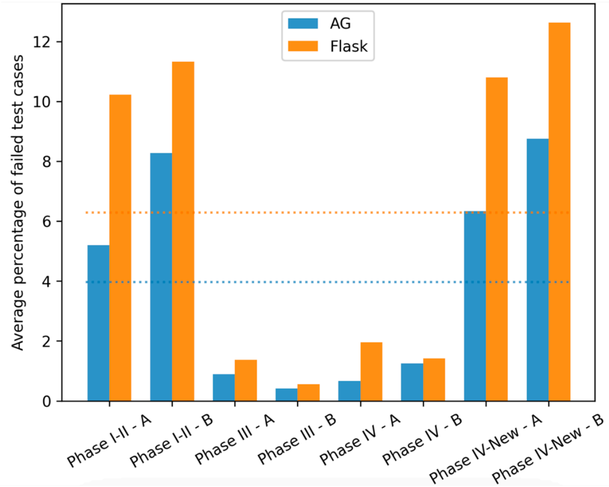
Time distribution (left) and average LoC per passed test (right) by phase and tool





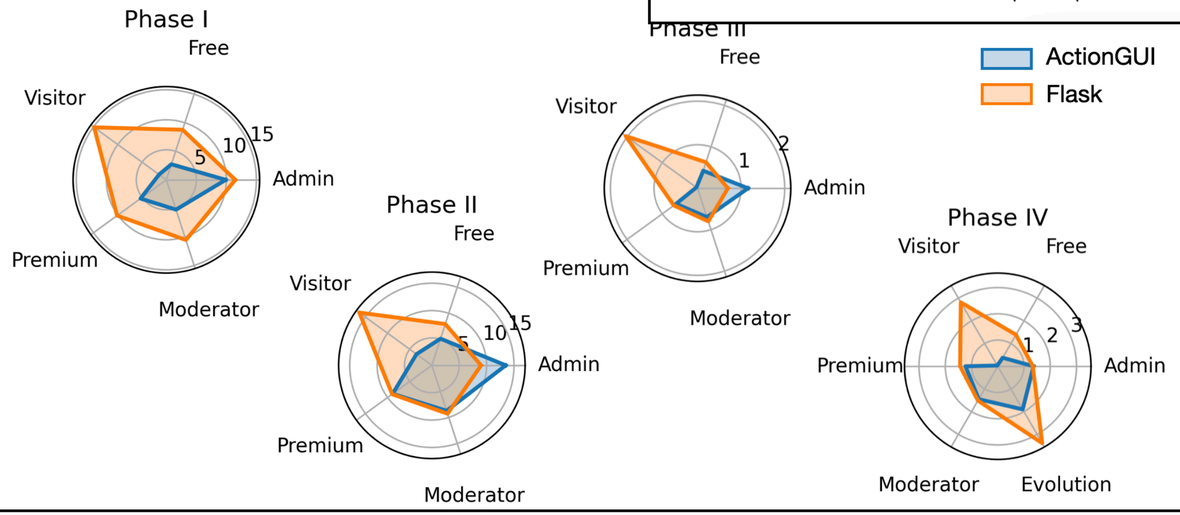
Results - Prior Knowledge [PK]

Average percentage of failed tests case by phase



Results - Difficulty [MD]

Average percentage of failed te



Conclusions and Future Work

- Empirically study on model-driven and code-centric approaches to AC.
  - Model-driven approach yields 50% fewer failed security tests.
  - Improvement carries over to repair and evolution.
  - The required effort is similar, but models are twice as concise.
- Larger studies (> 100 participants).
- Investigate whether model analysis improves specification of LP.
- Consider privacy policies

Is Modeling Access Control Worth It?

David Basin[1] Juan Guarnizo[2] Srđan Krstić[1] Hoang Nguyen[1] Martín Ochoa[2]

1

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

2

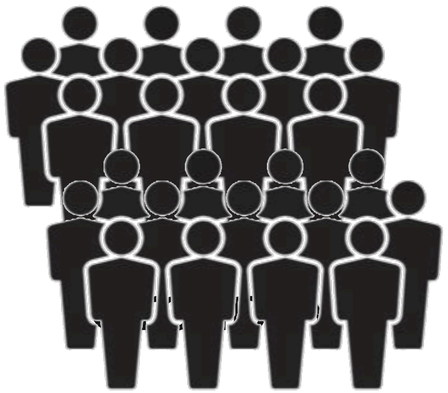
Zürcher Hochschule
für Angewandte Wissenschaften

zh aw School of Engineering

CCS 2023
Copenhagen, Denmark

Backup slides

Data Analysis



Students

	Code analysis	Security tests	Evolved security tests	Manual analysis
Initial models & implementations	[LoC]	[LP, CM, PK, MD]	N/A	[MD]
Repaired models & implementations	[LoC]	[LP, CM, PK, MD]	N/A	[MD]
Evolved models & implementations	[LoC]	[LP, CM, PK, MD]	[LP, CM, PK, MD]	[MD]
Anonymous answers	N/A	N/A	N/A	[T, MD]

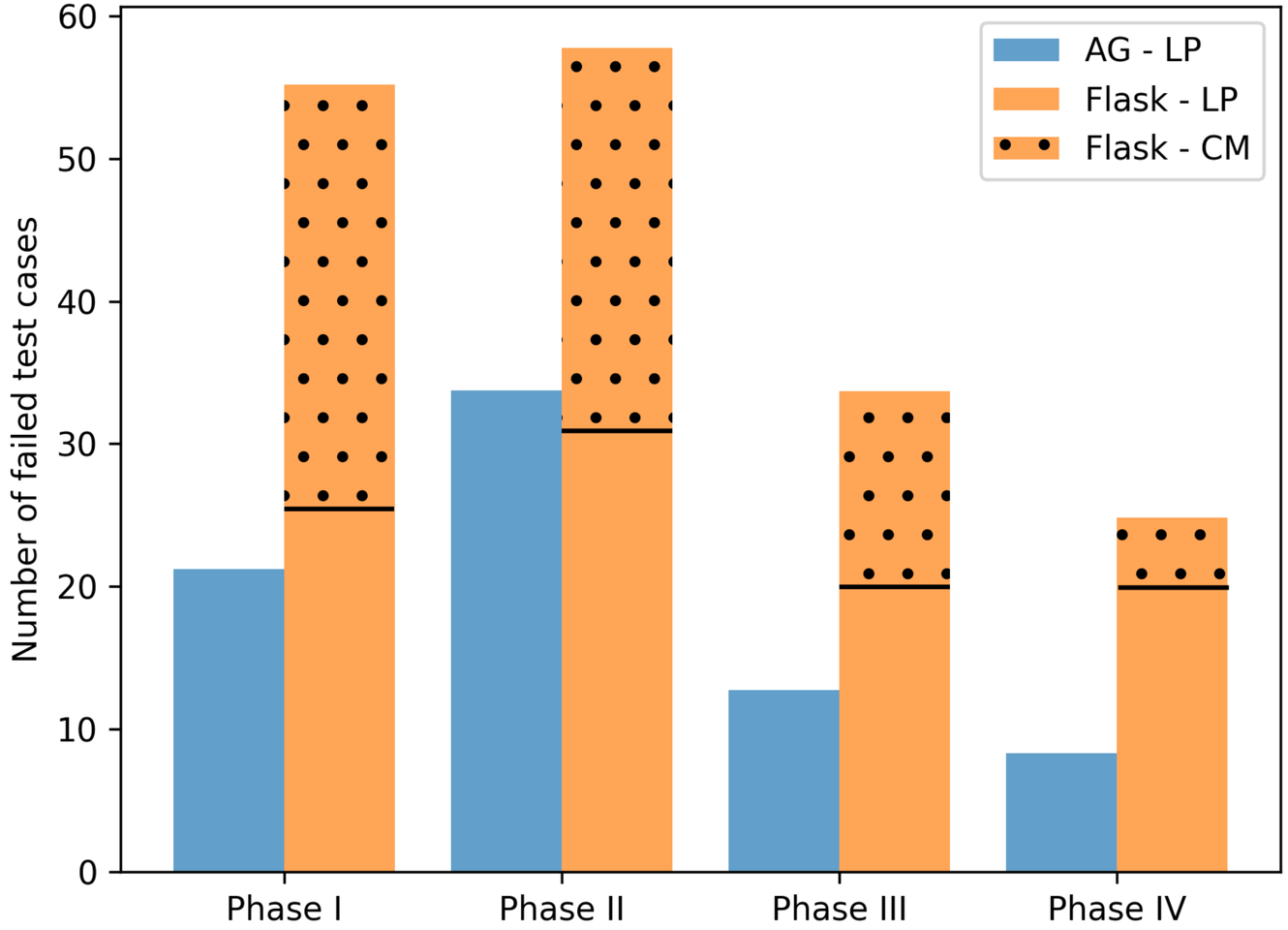
Results - Overview

Number of failed cases by tool and phase, averaged over submissions

		Initial Project		Evolved Project	
	Total submissions	**ActionGUI**	**Flask**	**ActionGUI**	**Flask**
Phase 1 (Implement)	75	4.23	12.41	N/A	N/A
Phase 2 (Switch)	74	6.75	11.20	N/A	N/A
Phase 3 (Repair)	69	0.51	1.00	N/A	N/A
Phase 4 (Evolve)	68	0.78	1.83	2.05	3.25

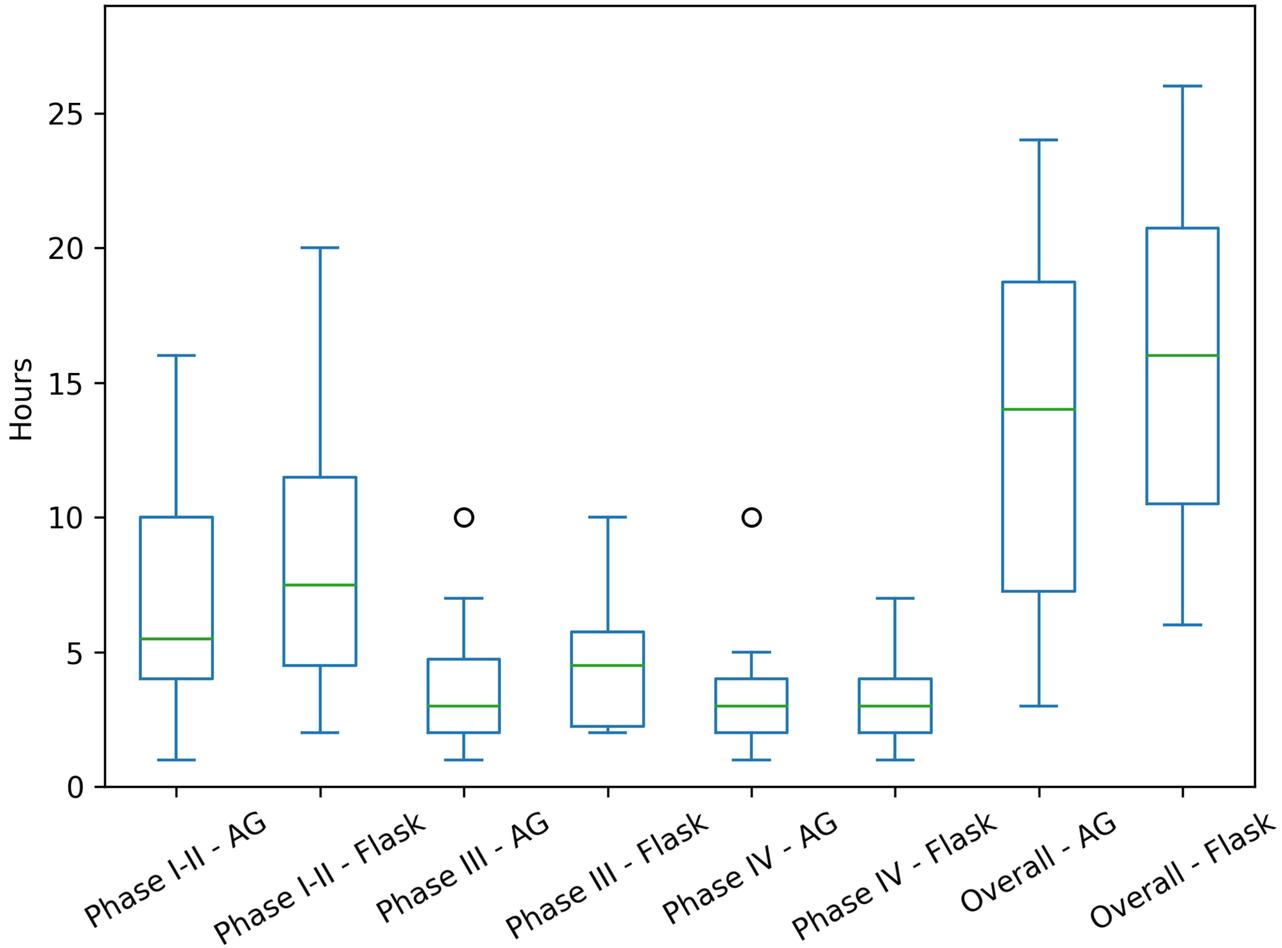
Results - Principles [LP & CM]

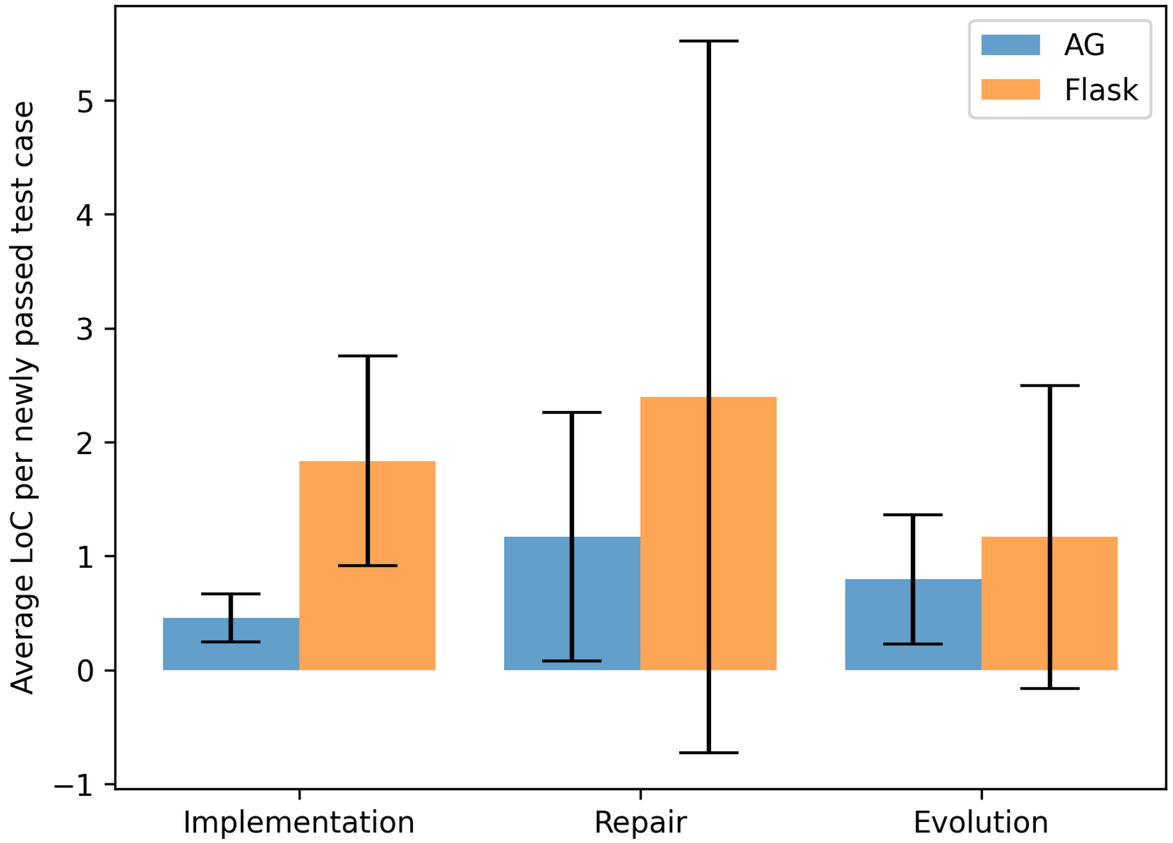
Number of failed cases by phase and violated principle (LP vs CM)



Results - Efficiency [T & LoC]

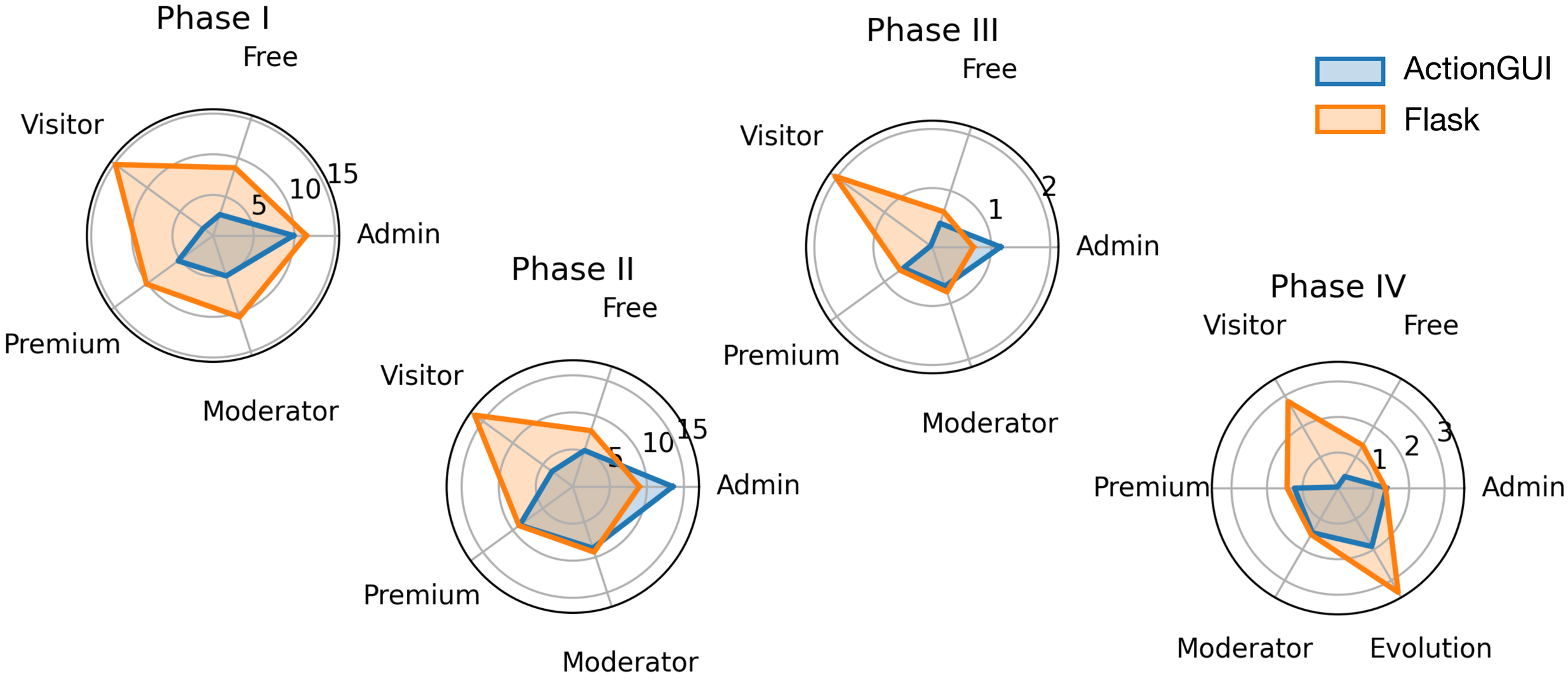
Time distribution (left) and average LoC per passed test (right) by phase and tool





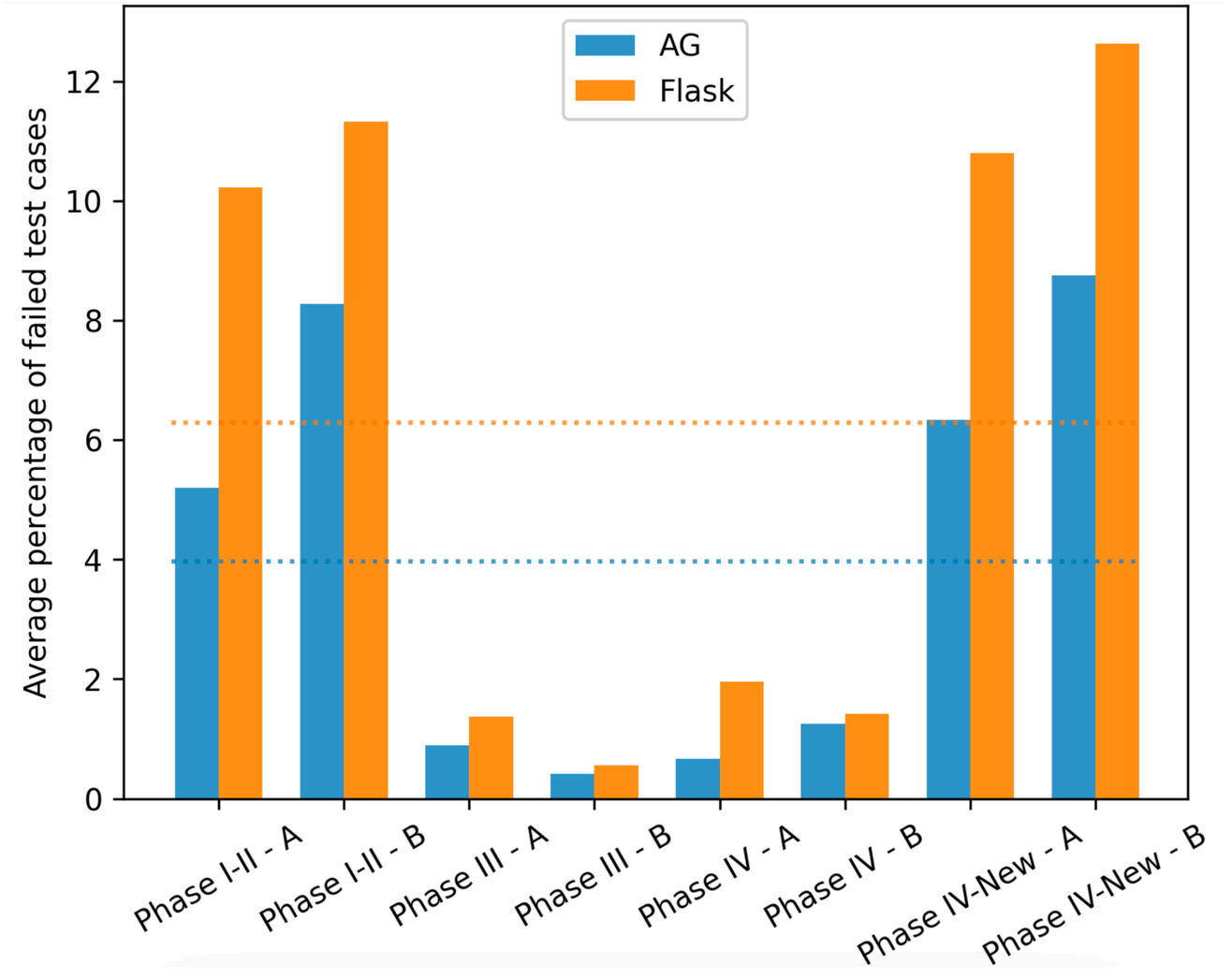
Results - Difficulty [MD]

Average percentage of failed tests per role and phase



Results - Prior Knowledge [PK]

Average percentage of failed tests case by phase



Related Work

- Domingo et al. [1]: MDD improves both efficiency and correct for video games; Roussev [5] shows that MDD improves learning goals in security courses
- Parkinson et al. meta-study [2]: Empirical studies on security of AC mechanisms mostly use synthetic data
- Gauthier et al. [3]: MDD improves evolution of AC policies based on structured interviews with experts
- BIBIFI [4]: “AC vulnerabilities are common”. Teams with detailed design made fewer errors
- Clavel et al. [6] report on lessons learned from applying MDD, specifically the potential for model re-usability and evolution
- Amazon’s CEDAR language: simple constraint language and no automatic CM

(References on the next page)

References

[1] África Domingo, Jorge Echeverría, Oscar Pastor, and Carlos Cetina. 2020. Evaluating the Benefits of Model-Driven Development - Empirical Evaluation Paper.

[2] Simon Parkinson and Saad Khan. 2023. A Survey on Empirical Security Analysis of Access-control Systems: A Real-world Perspective.

[3] François Gauthier, Ettore Merlo, Eleni Stroulia, and David Turner. 2014. Supporting Maintenance and Evolution of Access Control Models in Web Applications.

[4] Andrew Ruef, Michael Hicks, James Parker, Dave Levin, Michelle L. Mazurek, and Piotr Mardziel. 2016. Build It, Break It, Fix It: Contesting Secure Development.

[5] Borislav Roussev. 2003. Empirical Evidence Justifying the Adoption of a Model- Based Approach in the Course Web Applications Development.

[6] Manuel Clavel, Viviane Torresda Silva, Christiano Braga, and Marina Egea. 2008. Model-Driven Security in Practice: An Industrial Experience.